Typographical Conventions for None

Normal Text

```
 1  LIST X
 2 ; THIS IS THE MODIFIED SEPTEMBER ATARI 400/800 COMPUTER OPERATING
 3 ; SYSTEM LISTING. MODIFIED TO ASSEMBLE ON THE MICROTEC CROSS
 4 ; ASSEMBLER.
 5 ; THIS VERSION IS THE ONE WHICH WAS BURNED INTO ROM.
 6 ; THERE IS A RESIDUAL PIECE OF CODE WHICH IS FOR LNBUG. THIS
 7 ; IS AT LOCATION $9000 WHICH IS NOT IN ROM.
 8 ;
 9 ; THIS IS THE REVISION B EPROM VERSION
10         .PAGE
11 ;
12 ;
13 ;       COLLEEN OPERATING SYSTEM EQUATE FILE
14 ;
15 ;       NTSC/PAL ASSEMBLY FLAG
16 ;
17 PALFLG  =       0           ;0 = NTSC   1 = PAL
18 ;
19 ;
20 ;       MODULE ORIGIN TABLE
21 ;
22 CHRORG  =       $E000       ;CHARACTER SET
23 VECTBL  =       $E400       ;VECTOR TABLE
24 VCTABL  =       $E480       ;RAM VECTOR INITIAL VALUE TABLE
25 CIOORG  =       $E4A6       ;CENTRAL I/O HANDLER
26 INTORG  =       $E6D5       ;INTERRUPT HANDLER
27 SIOORG  =       $E944       ;SERIAL I/O DRIVER
28 DSKORG  =       $EDEA       ;DISK HANDLER
29 PRNORG  =       $EE78       ;PRINTER HANDLER
30 CASORG  =       $EF41       ;CASSETTE HANDLER
31 MONORG  =       $F0E3       ;MONITOR/POWER UP MODULE
32 KBDORG  =       $F3E4       ;KEYBOARD/DISPLAY HANDLER
33 ;
34 ;
35 ;
36 ;
37 ;      VECTOR TABLE
38 ;
39 ;HANDLER ENTRY POINTS ARE CALLED OUT IN THE FOLLOWING VECTOR
40 ;TABLE. THESE ARE THE ADDRESSES MINUS ONE.
41 ;
42 ;
43 ;EXAMPLE FOR EDITOR
44 ;
45 ;    E400       OPEN
46 ;       2       CLOSE
47 ;       4       GET
48 ;       6       PUT
49 ;       8       STATUS
50 ;       A       SPECIAL
51 ;       C       JUMP TO POWER ON INITIALIZATION ROUTINE
52 ;       F       NOT USED
53 ;
54 ;
55 EDITRV  =       $E400       ;EDITOR
```

```
 56 SCRENV  =       $E410       ;TELEVISION SCREEN
 57 KEYBDV  =       $E420       ;KEYBOARD
 58 PRINTV  =       $E430       ;PRINTER
 59 CASETV  =       $E440       ;CASSETTE
 60 ;
 61 ;       JUMP VECTOR TABLE
 62 ;
 63 ;THE FOLLOWING IS A TABLE OF JUMP INSTRUCTIONS
 64 ;TO VARIOUS ENTRY POINTS IN THE OPERATING SYSTEM.
 65 ;
 66 DISKIW  =       $E450       ;DISK INITIALIZATION
 67 DSKINV  =       $E453       ;DISK INTERFACE
 68 CIOV    =       $E456       ;CENTRAL INPUT OUTPUT ROUTINE
 69 SIOV    =       $E459       ;SERIAL INPUT OUTPUT ROUTINE
 70 SETVBV  =       $E45C       ;SET SYSTEM TIMERS ROUTINE
 71 SYSVBV  =       $E45F       ;SYSTEM VERTICAL BLANK CALCULATIONS
 72 XITVBV  =       $E462       ;EXIT VERTICAL BLANK CALCULATIONS
 73 SIOINV  =       $E465       ;SERIAL, INPUT OUTPUT INITIALIZATION
 74 SENDEV  =       $E468       ;SEND ENABLE ROUTINE
 75 INTINV  =       $E46B       ;INTERRUPT HANDLER INITIALIZATION
 76 CIOINV  =       $E46E       ;CENTRAL INPUT OUTPUT INITIALIZATION
 77 BLKBDV  =       $E471       ;BLACKBOARD MODE
 78 WARMSV  =       $E474       ;WARM START ENTRY POINT
 79 COLDSV  =       $E477       ;COLD START ENTRY POINT
 80 RBLOKV  =       $E47A       ;CASSETTE READ BLOCK ENTRY POINT VECTOR
 81 CSOPIV  =       $E47D       ;CASSETTE OPEN FOR INPUT VECTOR
 82 ;VCTABL = $E480
 83 ;
 84 ;
 85 ; OPERATING SYSTEM EQUATES
 86 ;
 87 ; COMMAND CODES FOR IOCB
 88 OPEN    =       3           ;OPEN FOR INPUT/OUTPUT
 89 GETREC  =       5           ;GET RECORD (TEXT)
 90 GETCHR  =       7           ;GET CHARACTER(S)
 91 PUTREC  =       9           ;PUT RECORD (TEXT)
 92 PUTCHR  =       $B          ;PUT CHARACTER(S)
 93 CLOSE   =       $C          ;CLOSE DEVICE
 94 STATIS  =       $D          ;STATUS REQUEST
 95 SPECIL  =       $E          ;BEGINNING OF SPECIAL ENTRY COMMANDS
 96 ;
 97 ; SPECIAL ENTRY COMMANDS
 98 DRAWLN  =       $11         ;DRAW LINE
 99 FILLIN  =       $12         ;DRAW LINE WITH RIGHT FILL
100 RENAME  =       $20         ;RENAME DISK FILE
101 DELETE  =       $21         ;DELETE DISK FILE
102 FORMAT  =       $22         ;FORMAT
103 LOCKFL  =       $23         ;LOCK FILE TO READ ONLY
104 UNLOCK  =       $24         ;UNLOCK LOCKED FILE
105 POINT   =       $25         ;POINT SECTOR
106 NOTE    =       $26         ;NOTE SECTOR
107 IOCFRE  =       $FF         ;IOCB "FREE"
108 ;
109 ; AUX1 EQUATES
110 ; () INDICATES WHICH DEVICES USE BIT
111 APPEND  =       $1          ;OPEN FOR WRITE APPEND (D), OR SCREEN READ (
112 DIRECT  =       $2          ;OPEN FOR DIRECTORY ACCESS (D)
113 OPNIN   =       $4          ;OPEN FOR INPUT (ALL DEVICES)
114 OPNOT   =       $8          ;OPEN FOR OUTPUT (ALL DEVICES)
```

```
115 OPNINO  =       OPNIN+OPNOT ;OPEN FOR INPUT AND OUTPUT (ALL DEVICES)
116 MXDMOD  =       $10         ;OPEN FOR MIXED MODE (E,S)
117 INSCLR  =       $20         ;OPEN WITHOUT CLEARING SCREEN (E,S)
118 ;
119 ; DEVICE NAMES
120 SCREDT  =       'E          ;SCREEN EDITOR (R/W)
121 KBD     =       'K          ;KEYBOARD (R ONLY)
122 D1SPLY  =       'S          ;SCREEN DISPLAY (R/W)
123 PRINTR  =       'P          ;PRINTER (W ONLY)
124 CASSET  =       'C          ;CASSETTE
125 MODEM   =       'M          ;MODEM
126 DISK    =       'D          ;DISK (R/W)
127 ;
128 ; SYSTEM EOL (CARRIAGE RETURN)
129 CR      =       $9B
130 ;
131 ;
132 ;       OPERATING SYSTEM STATUS CODES
133 ;
134 SUCCES  =       $01         ;SUCCESSFUL OPERATION
135 ;
136 BRKABT  =       $80         ;BREAK KEY ABORT
137 PRVOPN  =       $81         ;IOCB ALREADY OPEN
138 NONDEV  =       $82         ;NON-EXISTANT DEVICE
139 WRONLY  =       $83         ;IOCB OPENED FOR WRITE ONLY
140 NVALID  =       $84         ;INVALID COMMAND
141 NOTOPN  =       $85         ;DEVICE OR FILE NOT OPEN
142 BADIOC  =       $86         ;INVALID IOCB NUMBER
143 RDONLY  =       $87         ;IOCB OPENED FOR READ ONLY
144 EOFERR  =       $88         ;END OF FILE
145 TRNRCD  =       $89         ;TRUNCATED RECORD
146 TIMOUT  =       $8A         ;PERIPHERAL DEVICE TIME OUT
147 DNACK   =       $8B         ;DEVICE DOES NOT ACKNOWLEDGE COMMAND
148 FRMERR  =       $8C         ;SERIAL BUS FRAMING ERROR
149 CRSROR  =       $8D         ;CURSOR OVERRANCE
150 OVRRUN  =       $8E         ;SERIAL BUS DATA OVERRUN
151 CHKERR  =       $8F         ;SERIAL BUS CHECKSUM ERROR
152 ;
153 DERROR  =       $90         ;PERIPHERAL DEVICE ERROR (OPERATION NOT COMP
154 BADMOD  =       $91         ;BAD SCREEN MODE NUMBER
155 FNCNOT  =       $92         ;FUNCTION NOT IMPLEMENTED IN HANDLER
156 SCRMEM  =       $93         ;INSUFICIENT MEMORY FOR SCREEN MODE
157 ;
158 ;
159 ;
160 ;
161 ;
162 ;
163 ;       PAGE ZERO RAM ASSIGNMENTS
164 ;
165         *=$0000
166 LINZBS: .RES    2           ;LINBUG RAM (WILL BE REPLACED BY MONITOR RAM
167 ;
168 ; THESE LOCATIONS ARE NOT CLEARED
169 CASINI: .RES    2           ;CASSETTE INIT LOCATION
170 RAMLO:  .RES    2           ;RAM POINTER FOR MEMORY TEST
171 TRAMSZ: .RES    1           ;TEMPORARY REGISTER FOR RAM SIZE
172 TSTDAT: .RES    1           ;RAM TEST DATA REGISTER
173 ;
```

```
174 ; CLEARED ON COLOSTART ONLY
175 WARMST: .RES    1           ;WARM START FLAG
176 BOOT?:  .RES    1           ;SUCCESSFUL BOOT FLAG
177 DOSVEC: .RES    2           ;DISK SOFTWARE START VECTOR
178 DOSINI: .RES    2           ;DISK SOFTWARE INIT ADDRESS
179 APPMHI: .RES    2           ;APPLICATIONS MEMORY HI LIMIT
180 ;
181 ; CLEARED ON COLD OR WARM START
182 INTZBS  =*                  ;INTERRUPT HANDLER
183 POKMSK: .RES    1           ;SYSTEM MASK FOR POKEY IRQ ENABLE
184 BRKKEY: .RES    1           ;BREAK KEY FLAG
185 RTCLOK: .RES    3           ;REAL TIME CLOCK (IN 16 MSEC UNITS)
186 ;
187 BUFADR: .RES    2           ;INDIRECT BUFFER ADDRESS REGISTER
188 ;
189 ICCOMT: .RES    1           ;COMMAND FOR VECTOR
190 ;
191 DSKFMS: .RES    2           ;DISK FILE MANAGER POINTER
192 DSKUTL: .RES    2           ;DISK UTILITIES POINTER
193 ;
194 PTIMOT: .RES    1           ;PRINTER TIME OUT REGISTER
195 PBPNT:  .RES    1           ;PRINT BUFFER POINTER
196 PBUFSZ: .RES    1           ;PRINT BUFFER SIZE
197 PTEMP:  .RES    1           ;TEMPORARY REGISTER
198 ;
199 ZIOCB   =*                  ;ZERO PAGE I/O CONTROL BLOCK
200 IOCBSZ  =       16          ;NUMBER OF BYTES PER IOCB
201 MAXIOC  =       8*IOCBSZ    ;LENGTH OF THE IOCB AREA
202 IOCBAS  =*
203 ICHIDZ: .RES    1           ;HANDLER INDEX NUMBER (FF = IOCB FREE)
204 ICDNOZ: .RES    1           ;DEVICE NUMBER (DRIVE NUMBER)
205 ICCOMZ: .RES    1           ;COMMAND CODE
206 ICSTAZ: .RES    1           ;STATUS OF LAST IOCB ACTION
207 ICBALZ: .RES    1           ;BUFFER ADDRESS LOW BYTE
208 ICBAHZ: .RES    1
209 ICPTLZ: .RES    1           ;PUT BYTE ROUTINE ADDRESS - 1
210 ICPTHZ: .RES    1
211 ICBLLZ: .RES    1           ;BUFFER LENGTH LOW BYTE
212 ICBLHZ: .RES    1
213 ICAX1Z: .RES    1           ;AUXILIARY INFORMATION FIRST BYTE
214 ICAX2Z: .RES    1
215 ICSPRZ: .RES    4           ;TWO SPARE BYTES (CIO LOCAL USE)
216 ICIDNO  =       ICSPRZ+2    ;IOCB NUMBER X 16
217 CIOCHR  =       ICSPRZ+3    ;CHARACTER BYTE FOR CURRENT OPERATION
218 ;
219 STATUS: .RES    1           ;INTERNAL STATUS STORAGE
220 CHKSUM: .RES    1           ;CHECKSUM (SINGLE BYTE SUM WITH CARRY)
221 BUFRLO: .RES    1           ;POINTER TO DATA BUFFER (LO BYTE)
222 BUFRHI: .RES    1           ;POINTER TO DATA BUFFER (HI BYTE)
223 BFENLO: .RES    1           ;NEXT BYTE PAST END OF THE DATA BUFFER (LO B
224 BFENHI: .RES    1           ;NEXT BYTE PAST END OF THE DATA BUFFER (HI B
225 CRETRY: .RES    1           ;NUMBER OF COMMAND FRAME RETRIES
226 DRETRY: .RES    1           ;NUMBER OF DEVICE RETRIES
227 BUFRFL: .RES    1           ;DATA BUFFER FULL FLAG
228 RECVDN: .RES    1           ;RECEIVE DONE FLAG
229 XMTDON: .RES    1           ;TRANSMISSION DONE FLAG
230 CHKSNT: .RES    1           ;CHECKSUM SENT FLAG
231 NOCKSM: .RES    1           ;NO CHECKSUM FOLLOWS DATA FLAG
232 ;
```

```
233 ;
234 BPTR:   .RES    1
235 FTYPE:  .RES    1
236 FEOF:   .RES    1
237 FREQ:   .RES    1
238 SOUNDR: .RES    1           ;NOISY I/O FLAG. (ZERO IS QUIET)
239 CRITIC: .RES    1           ;DEFINES CRITICAL SECTION (CRITICAL IF NON-Z
240 ;
241 FMSZPG: .RES    7           ;DISK FILE MANAGER SYSTEM ZERO PAGE
242 ;
243 ;
244 CKEY:   .RES    1           ;FLAG SET WHEN GAME START PRESSED
245 CASSBT: .RES    1           ;CASSETTE BOOT FLAG
246 DSTAT:  .RES    1           ;DISPLAY STATUS
247 ;
248 ATRACT: .RES    1           ;ATRACT FLAG
249 DRKMSK: .RES    1           ;DARK ATRACT MASK
250 COLRSH: .RES    1           ;ATRACT COLOR SHIFTER (EOR'ED WITH PLAYFIELD
251 ;
252 LEDGE   =       2           ;LMARGN'S VALUE AT COLD START
253 REDGE   =       39          ;RMARGN'S VALUE AT COLD START
254 TMPCHR: .RES    1
255 HOLD1:  .RES    1
256 LMARGN: .RES    1           ;LEFT MARGIN (SET TO 1 AT POWER ON)
257 RMARGN: .RES    1           ;RIGHT MARGIN (SET TO 38 AT POWER ON)
258 ROWCRS: .RES    1           ;CURSOR COUNTERS
259 COLCRS: .RES    2
260 DINDEX: .RES    1
261 SAVMSC: .RES    2
262 OLDROW: .RES    1
263 OLDCOL: .RES    2
264 OLDCHR: .RES    1           ;DATA UNDER CURSOR
265 OLDADR: .RES    2
266 NEWROW: .RES    1           ;POINT DRAW GOES TO
267 NEWCOL: .RES    2
268 LOGCOL: .RES    1           ;POINTS AT COLUMN IN LOGICAL LINE
269 ADRESS: .RES    2
270 MLTTMP: .RES    2
271 OPNTMP  =       MLTTMP      ;FIRST BYTE IS USED IN OPEN AS TEMP
272 SAVADR: .RES    2
273 RAMTOP: .RES    1           ;RAM SIZE DEFINED BY POWER ON LOGIC
274 BUFCNT: .RES    1           ;BUFFER COUNT
275 BUFSTR: .RES    2           ;EDITOR GETCH POINTER
276 BITMSK: .RES    1           ;BIT MASK
277 SHFAMT: .RES    1
278 ROWAC:  .RES    2
279 COLAC:  .RES    2
280 ENDPT:  .RES    2
281 DELTAR: .RES    1
282 DELTAC: .RES    2
283 ROWINC: .RES    1
284 COLINC: .RES    1
285 SWPFLG: .RES    1           ;NON-0 IF TXT AND REGULAR RAM IS SWAPPED
286 HOLDCH: .RES    1           ;CH IS MOVED HERE IN KGETCH BEFORE CNTL & SH
287 INSDAT: .RES    1
288 COUNTR: .RES    2
289 ;
290 ;
291 ;
```

```
292 ;
293 ;       30 - FF ARE RESERVED FOR USER APPLICATIONS
294 ;
295 ;
296 ;
297 ;       NOTE : SEE FLOATING POINT SUBROUTINE AREA FOR ZERO PAGE CELLS
298 ;
299 ;
300 ;
301 ;
302 ;       PAGE 1   -    STACK
303 ;
304 ;
305 ;
306 ;
307 ;       PAGE TWO RAM ASSIGNMENTS
308 ;
309         *=$0200
310 INTABS  =*                  ;INTERRUPT RAM
311 VDSLST: .RES    2           ;DISPLAY LIST NMI VECTOR
312 VPRCED: .RES    2           ;PROCEED LINE IRQ VECTOR
313 VINTER: .RES    2           ;INTERRUPT LINE IRQ VECTOR
314 VBREAK: .RES    2           ;SOFTWARE BREAK (00) INSTRUCTION IRQ VECTOR
315 VKEYBD: .RES    2           ;POKEY KEYBOARD IRQ VECTOR
316 VSERIN: .RES    2           ;POKEY SERIAL INPUT READY IRQ
317 VSEROR: .RES    2           ;POKEY SERIAL OUTPUT READY IRQ
318 VSEROC: .RES    2           ;POKEY SERIAL OUTPUT COMPLETE IRQ
319 VTIMR1: .RES    2           ;POKEY TIMER 1 IRQ
320 VTIMR2: .RES    2           ;POKEY TIMER 2 IRQ
321 VTIMR4: .RES    2           ;POKEY TIMER 4 IRQ
322 VIMIRU: .RES    2           ;IMMEDIATE IRQ VECTOR
323 CDTMV1: .RES    2           ;COUNT DOWN TIMER 1
324 CDTMV2: .RES    2           ;COUNT DOWN TIMER 2
325 CDTMV3: .RES    2           ;COUNT DOWN TIMER 3
326 CDTMV4: .RES    2           ;COUNT DOWN TIMER 4
327 CDTMV5: .RES    2           ;COUNT DOWN TIMER S
328 VVBLKI: .RES    2           ;IMMEDIATE VERTICAL BLANK NMI VECTOR
329 VVBLKD: .RES    2           ;DEFERRED VERTICAL BLANK NMI VECTOR
330 CDTMA1: .RES    2           ;COUNT DOWN TIMER 1 JSR ADDRESS
331 CDTMA2: .RES    2           ;COUNT DOWN TIMER 2 JSR ADDRESS
332 CDTMF3: .RES    1           ;COUNT DOWN TIMER 3 FLAG
333 SRTIMR: .RES    1           ;SOFTWARE REPEAT TIMER
334 CDTMF4: .RES    1           ;COUNT DOWN TIMER 4 FLAG
335 INTEMP: .RES    1           ;IAN'S TEMP (RENAMED FROM T1 BY POPULAR DEMA
336 CDTMF5: .RES    1           ;COUNT DOWN TIMER FLAG 5
337 SDMCTL: .RES    1           ;SAVE DMACTL REGISTER
338 SDLSTL: .RES    1           ;SAVE DISPLAY LIST LOW BYTE
339 SDLSTH: .RES    1           ;SAVE DISPLAY LIST HI BYTE
340 SSKCTL: .RES    1           ;SKCTL REGISTER RAM
341         .RES    1
342 ;
343 LPENH:  .RES    1           ;LIGHT PEN HORIZONTAL VALUE
344 LPENV:  .RES    1           ;LIGHT PEN VERTICAL VALUE
345 BRKKY:  .RES    2           ;BREAK KEY VECTOR
346 ;
347         .RES    2           ;SPARE
348 ;
349 CDEVIC: .RES    1           ;COMMAND FRAME BUFFER - DEVICE
350 CCOMND: .RES    1           ;COMMAND
```

```
351 CAUX1:  .RES    1           ;COMMAND AUX BYTE 1
352 CAUX2:  .RES    1           ;COMMANDAUX BYTE 2
353 ;   NOTE: MAY NOT BE THE LAST WORD ON A PAGE
354 TEMP:   .RES    1           ;TEMPORARY RAM CELL
355 ;   NOTE: MAY NOT BE THE LAST WORD ON A PAGE
356 ERRFLG: .RES    1           ;ERROR FLAG - ANY DEVICE ERROR EXCEPT TIME
357 ;
358 DFLAGS: .RES    1           ;DISK FLAGS FROM SECTOR ONE
359 DBSECT: .RES    1           ;NUMBER OF DISK BOOT SECTORS
360 BOOTAD: .RES    2           ;ADDRESS WHERE DISK BOOT LOADERWILL BE PUT
361 COLDST: .RES    1           ;COLDSTART FLAG (1=IN MIDDLE OF COLDSTART)
362 ;
363         .RES    1           ;SPARE
364 ;
365 DSKTIM: .RES    1           ;DISK TIME OUT REGISTER
366 ;
367 LINBUF: .RES    40          ;CHAR LINE BUFFER
368 ;
369 GPRIOR: .RES    1           ;GLOBAL PRIORITY CELL
370 ;
371 PADDL0: .RES    1           ;POTENTIOMETER 0 RAM CELL
372 PADDL1: .RES    1
373 PADDL2: .RES    1
374 PAODL3: .RES    1
375 PADDL4: .RES    1
376 PADDL5: .RES    1
377 PADDL6: .RES    1
378 PADDL7: .RES    1
379 STICK0: .RES    1           ;JOYSTICK 0 RAM CELL
380 STICK1: .RES    1
381 STICK2: .RES    1
382 STICK3: .RES    1
383 PTRIG0: .RES    1           ;PADDLE TRIGGER 0
384 PTRIG1: .RES    1
385 PTRIG2: .RES    1
386 PTRIG3: .RES    1
387 PTRIG4: .RES    1
388 PTRIG5: .RES    1
389 PTRIG6: .RES    1
390 PTRIG7: .RES    1
391 STRIG0: .RES    1           ;JOYSTICK TRIGGER 0
392 STRIG1: .RES    1
393 STRIG2: .RES    1
394 STRIG3: .RES    1
395 ;
396 CSTAT:  .RES    1
397 WMODE:  .RES    1
398 BLIM:   .RES    1
399 IMASK:  .RES    1
400 JVECK:  .RES    2
401 ;
402         .RES    2           ;SPARE
403 ;
404 ;
405 ;
406 ;
407 TXTROW: .RES    1           ;TEXT ROWCRS
408 TXTCOL: .RES    2           ;TEXT COLCRS
409 TINDEX: .RES    1           ;TEXT INDEX
```

```
410 TXTMSC: .RES    2           ;FOOLS CONVRT INTO NEW MSC
411 TXTOLD: .RES    6           ;OLDROW & OLDCOL FOR TEXT (AND THEN SOME)
412 TMPX1:  .RES    1
413 HOLD3:  .RES    1
414 SUBTMP: .RES    1
415 HOLD2:  .RES    1
416 DMASK:  .RES    1
417 TMPLBT: .RES    1
418 ESCFLG: .RES    1           ;ESCAPE FLAG
419 TABMAP: .RES    15
420 LOGMAP: .RES    4           ;LOGICAL LINE START BIT MAP
421 INVFLG: .RES    1           ;INVERSE VIDEO FLAG (TOGGLED BY ATARI KEY)
422 FILFLG: .RES    1           ;RIGHT FILL FLAG FOR DRAW
423 TMPROW: .RES    1
424 TMPCOL: .RES    2
425 SCRFLG: .RES    1           ;SET IF SCROLL OCCURS
426 HOLD4:  .RES    1           ;TEMP CELL USED IN DRAW ONLY
427 HOLD5:  .RES    1           ;DITTO
428 SHFLOK: .RES    1
429 BOTSCR: .RES    1           ;BOTTOM OF SCREEN : 24 NORM 4 SPLIT
430 ;
431 ;
432 PCOLR0: .RES    1           ;P0 COLOR
433 PCOLR1: .RES    1           ;P1 COLOR
434 PCOLR2: .RES    1           ;P2 COLOR
435 PCOLP3: .RES    1           ;P3 COLOR
436 COLOR0: .RES    1           ;COLOR 0
437 COLOR1: .RES    1
438 CQLOR2: .RES    1
439 COLOR3: .RES    1
440 COLOR4: .RES    1
441 ;
442 ;
443         .RES    23          ;SPARE
444 ;
445 ;
446 ;
447 GLBABS  =*                  ;GLOBAL VARIABLES
448 ;
449         .RES    4           ;SPARE
450 ;
451 RAMSIZ: .RES    1           ;RAM SIZE (HI BYTE ONLY)
452 MEMTOP: .RES    2           ;TOP OF AVAILABLE USER MEMORY
453 MEMLO:  .RES    2           ;BOTTOM OF AVAILABLE USER MEMORY
454         .RES    1           ;SPARE
455 DVSTAT: .RES    4           ;STATUS BUFFER
456 CBAUDL: .RES    1           ;CASSETTE BAUD RATE LOW BYTE
457 CBAUDH: .RES    1
458 ;
459 CRSINH: .RES    1           ;CURSOR INHIBIT (00 = CURSOR ON)
460 KEYDEL: .RES    1           ;KEY DELAY
461 CH1:    .RES    1
462 ;
463 CHACT:  .RES    1           ;CHACTL REGISTER RAM
464 CHBAS:  .RES    1           ;CHBAS REGISTER RAM
465 ;
466         .RES    5           ;SPARE BYTES
467 ;
468 CHAR:   .RES    1
```

```
469 ATACHR: .RES    1           ;ATASCII CHARACTER
470 CH:     .RES    1           ;GLOBAL VARIABLE FOR KEYBOARD
471 FILDAT: .RES    1           ;RIGHT FILL DATA (DRAW)
472 DSPFLG: .RES    1           ;DISPLAY FLAG : DISPLAY CNTLS IF NON-ZERO
473 SSFLAG: .RES    1           ;START/STOP FLAG FOR PAGING (CNTL 1). CLEARE
474 ;
475 ;
476 ;
477 ;
478 ;
479 ;
480 ;
481 ;       PAGE THREE RAM ASSIGNMENTS
482 ;
483 DCB     =*          ;DEVICE CONTROL BLOCK
484 DDEVIC: .RES    1           ;PERIPHERAL UNIT 1 BUS ID. NUMBER
485 DUNIT:  .RES    1           ;UNIT NUMBER
486 DCOMND: .RES    1           ;BUS COMMAND
487 DSTATS: .RES    1           ;COMMAND TYPE/STATUS RETURN
488 DBUFLO: .RES    1           ;DATA BUFFER POINTER LOW BYTE
489 DBUFHI: .RES    1
490 DTIMLO: .RES    1           ;DEVICE TIME OUT IN 1 SECOND UNITS
491 DUNUSE: .RES    1           ;UNUSED BYTE
492 DBYTLO: .RES    1           ;NUMBER OF BYTES TO BE TRANSFERRED LOW BYTE
493 DBYTHI: .RES    1
494 DAUX1:  .RES    1           ;COMMAND AUXILIARY BYTE 1
495 DAUX2:  .RES    1
496 ;
497 TIMER1: .RES    2           ;INITIAL TIMER VALUE
498 ADDCOR: .RES    1           ;ADDITION CORRECTION
499 CASFLG: .RES    1           ;CASSETTE MODE WHEN SET
500 TIMER2: .RES    2           ;FINAL TIMER VALUE. THESE TWO TIMER VALUES
501 ; ARE USED TO COMPUTE INTERVAL FOR BAUD RATE
502 TEMP1:  .RES    2           ;TEMPORARY STORAGE REGISTER
503 TEMP2:  .RES    1           ;TEMPORARY STORAGE REGISTER
504 TEMP3:  .RES    1           ;TEMPORARY STORAGE REGISTER
505 SAVIO:  .RES    1           ;SAVE SERIAL IN DATA PORT
506 TIMFLG: .RES    1           ;TIME OUT FLAG FOR BAUD RATE CORRECTION
507 STACKP: .RES    1           ;SIO STACK POINTER SAVE CELL
508 TSTAT:  .RES    1           ;TEMPORARY STATUS HOLDER
509 ;
510 ;
511 ;
512 HATABS: .RES    38          ;HANDLER ADDRESS TABLE
513 MAXDEV  =       *-HATABS-5  ;MAXIMUM HANDLER ADDRESS INDEX
514 ;
515 ;     NOTE : THE ENTIRE IOCB DEFINITIONS HAVE BEEN MODIFIED
516 ;
517 IOCB:   .ORG    *           ;I/O CONTROL BLOCKS
518 ICHID:  .RES    1           ;HANDLER INDEX NUMBER (FF = IOCB FREE)
519 ICDNO:  .RES    1           ;DEVICE NUMBER (DRIVE NUMBER)
520 ICCOM:  .RES    1           ;COMMAND CODE
521 ICSTA:  .RES    1           ;STATUS OF LAST IOCB ACTION
522 ICBAL:  .RES    1           ;BUFFER ADDRESS LOW BYTE
523 ICBAH:  .RES    1
524 ICPTL:  .RES    1           ;PUT BYTE ROUTINE ADDRESS - 1
525 ICPTH:  .RES    1
526 ICBLL:  .RES    1           ;BUFFER LENGTH LOW BYTE
527 ICBLH:  .RES    1
```

```
528 ICAX1:  .RES    1           ;AUXILIARY INFORMATION FIRST BYTE
529 ICAX2:  .RES    1
530 ICSPR:  .RES    4           ;FOUR SPARE BYTES
531         .RES    MAXIOC-IOCBSZ
532 ;
533 PRNBUF: .RES    40          ;PRINTER BUFFER
534 ;
535         .RES    21          ;SPARE BYTES
536 ;
537 ;
538 ;
539 ;
540 ;
541 ;
542 ;
543 ;       PAGE FOUR RAM ASSIGNMENTS
544 ;
545 CASBUF: .RES    131         ;CASSETTE BUFFER
546 ;
547 ; USER AREA STARTS HERE AND GOES TO END OF PAGE FIVE
548 USAREA: .RES    128         ;SPARE
549 ;
550 ;
551 ;
552 ;
553 ;
554 ;
555 ;
556 ;       PAGE FIVE RAM ASSIGNMENTS
557 ;
558 ;       PAGE FIVE IS RESERVED AS A USER WORK SPACE
559 ;
560 ;       NOTE:   SEE FLOATING POINT SUBROUTINE AREA FOR PAGE FIVE CELLS
561 ;
562 ;
563 ;       PAGE SIX RAM ASSIGNMENTS
564 ;
565 ; PAGE SIX IS RESERVED AS A USER'S USER WORK SPACE
566 ;
567 ;
568 ;
569 ;
570 ;       FLOATING POINT SUBROUTINES
571 ;
572 FPREC   =       6           ;FLOATING PT PRECISION (# OF BYTES)
573 ; IF CARRY USED THEN CARRY CLEAR => NO ERROR, CARR
574 AFP     =       $D800       ;ASCII->FLOATING POINT(FP)
575 ;                               INBUFF+CIX -> FR0, CIX, CARRY
576 FASC    =       $D8E6       ;FP -> ASCII FR0 -> LBUFF (INBUFF)
577 IFP     =       $D9AA       ;INTEGER -> FP
578 ;                               0-$FFFF (LSB,MSB) IN FR0,FR0+1->FR0
579 FPI     =       $D9D2       ;FP -> INTEGER FR0 -> FR0,FR0+1, CARRY
580 FSUB    =       $DA60       ;FR0 <- FR0 - FR1 ,CARRY
581 FADD    =       $DA66       ;FR0 <- FR0 + FR1 ,CARRY
582 FMUL    =       $DADB       ;FR0 <- FR0 * FR1 ,CARRY
583 FDIV    =       $DB28       ;FR0 <- FR0 / FR1 ,CARRY
584 FLD0R   =       $DD89       ;FLOATING LOAD REG0   FR0  <- (X,Y)
585 FLD0P   =       $DD80       ;   "       "    "     FR0  <- (FLFTR)
586 FLD1R   =       $DD98       ;   "       "   REG1   FR1  <- (X,Y)
```

```
587 FLD1P   =       $DD9C       ;   "       "    "     FR1  <- (FLPTR)
588 FSTOR   =       $DDA7       ;FLOATING STORE REG0 (X,Y) <- FR0
589 FSTOP   =       $DDAB       ;    "      "    " (FLPTR)  <- FR0
590 FMOVE   =       $DDB6       ;FR1 <- FR0
591 PLYEVL  =       $DD40       ;FR0 <- P(Z) = SUM(I=N TO 0) (A(I)*Z**I) CAR
592 ;                           INPUT:  (X,Y) = A(N),A(N-1)...A(0) -> PLYARG
593 ;                                   ACC   = # OF COEFFICIENTS = DEGREE+1
594 ;                                   FR0   = Z
595 EXP     =       $DDC0       ;FR0 <- E**FR0 = EXP10(FR0 * LOG10(E)) CARRY
596 EXP10   =       $DDCC       ;FR0 <- 10**FR0 CARRY
597 LOG     =       $DECD       ;FR0 <- LN(FR0) = LOG10(FR0)/LOG10(E) CARRY
598 LOG10   =       $DED1       ;FR0 <- LOG10 (FR0) CARRY
599 ; THE FOLLOWING ARE IN BASIC CARTRIDGE:
600 SIN     =       $BDB1       ;FR0 <- SIN(FR0) DEGFLG=0 =>RADS, 6=>DEG. CA
601 COS     =       $BD73       ;FR0 <- COS(FR0) CARRY
602 ATAN    =       $BE43       ;FR0 <- ATAN(FR0) CARRY
603 SQR     =       $BEB1       ;FR0 <- SQUAREROOT(FR0) CARRY
604 ; FLOATING POINT ROUTINES ZERO PAGE (NEEDED ONLY IF V.P. ROUTINES ARE CA
605         *=$D4
606 FR0:    .RES    FPREC       ;FP REG0
607 FRE:    .RES    FPREC
608 FR1:    .RES    FPREC       ;FP REG1
609 FR2:    .RES    FPREC
610 FRX:    .RES    1           ;FP SPARE
611 EEXP:   .RES    1           ;VALUE OF E
612 NSIGN:  .RES    1           ;SIGN OF #
613 ESIGN:  .RES    1           ;SIGN OF EXPONENT
614 FCHRFLG:.RES    1           ;1ST CHAR FLAG
615 DIORT:  .RES    1           ;# OF DIGITS RIGHT OF DECIMAL
616 CIX:    .RES    1           ;CURRENT INPUT INDEX
617 INBUFF: .RES    2           ;POINTS TO USER'S LINE INPUT BUFFER
618 ZTEMPI: .RES    2
619 ZIEMP4: .RES    2
620 ZTEMP3: .RES    2
621 DEGFLG
622 RADFLG: .RES    1           ;0=RADIANS, 6=DEGREES
623 RADON   =       0           ;INDICATES RADIANS
624 DEGON   =       6           ;INDICATES DEGREES
625 FLPTR:  .RES    2           ;POINTS TO USER'S FLOATING PT NUMBER
626 FPTR2:  .RES    2
627 ; FLOATING PT ROUTINES' NON-ZERO PAGE RAM
628 ; (NEEDED ONLY IF F.P. ROUTINES CALLED)
629         *=$57E
630 LBPR1:  .RES    1           ;LBUFF PREFIX 1
631 LBPR2:  .RES    1           ;LBUFF PREFIX 2
632 LBUFF:  .RES    128         ;LINE BUFFER
633 PLYARG  =       LBUFF+$60   ;POLYNOMIAL ARGUMENTS
634 FPSCR   =       PLYARG+FPREC
635 FPSCR1  =       FPSCR+FPREC
636 FSCR    =       FPSCR
637 FSCR1   =       FPSCR1
638 LBFEND  =       *-1         ;END OF LBUFF
639 ;
640 ;
641 ;
642 ;
643 ;
644 ;
645 ;
```

```
646 ;
647 ;
648 ;       COLLEEN MNEMONICS
649 ;
650 POKEY   =       $D200       ;VBLANK ACTION:           DESCRIPTION:
651 POT0    =       POKEY+0     ;POT0-->PADDL0           0-227 IN RAM CELL
652 POT1    =       POKEY+1     ;POT1-->PADDL1           0-227 IN RAM CELL
653 POT2    =       POKEY+2     ;POT2-->PADDL2           0-227 IN RAM CELL
654 POT3    =       POKEY+3     ;POT3-->PADDL3           0-227 IN RAM CELL
655 POT4    =       POKEY+4     ;POT4-->PADDL4           0-227 IN RAM CELL
656 POT5    =       POKEY+5     ;POT5-->PADDL5           0-227 IN RAM CELL
657 POT6    =       POKEY+6     ;POT6-->PADDL6           0-227 IN RAM CELL
658 POT7    =       POKEY+7     ;POT7-->PADDL7           0-227 IN RAM CELL
659 ALLPOT  =       POKEY+8     ;???
660 KBCODE  =       POKEY+9
661 RANDOM  =       POKEY+10
662 POTGO   =       POKEY+11    ;STROBED
663 SERIN   =       POKEY+13
664 IRQST   =       POKEY+14
665 SKSTAT  =       POKEY+15
666 AUDF1   =       POKEY+0
667 AUDC1   =       POKEY+1
668 AUDF2   =       POKEY+2
669 AUDC2   =       POKEY+3
670 AUDF3   =       POKEY+4
671 AUDC3   =       POKEY+5
672 AUDF4   =       POKEY+6
673 AUDC4   =       POKEY+7
674 AUDCTL  =       POKEY+8     ;NONE               AUDCTL<--[SIO]
675 STIMER  =       POKEY+9
676 SKRES   =       POKEY+10    ;NONE                SKRES<--[SIO]
677 SEROUT  =       POKEY+13    ;NONE               SEROUT<--[SIO]
678 IRQEN   =       POKEY+14    ;POKMSK-->IRQEN (AFFECTED BY OPEN S: OR E:)
679 SKCTL   =       POKEY+15    ;SSKCTL-->SKCTL      SSKCTL<--[SIO]
680 ;
681 CTIA    =       $D000       ;VBLANK ACTION:       DESCRIPTION:
682 HPOSP0  =       CTIA+0
683 HPOSP1  =       CTIA+1
684 HPOSP2  =       CTIA+2
685 HPOSP3  =       CTIA+3
686 HPOSM0  =       CTIA+4
687 HPOSM1  =       CTIA+5
688 HPOSM2  =       CTIA+6
689 HPOSM3  =       CTIA+7
690 SIZEP0  =       CTIA+8
691 SIZEP1  =       CTIA+9
692 SIZEP2  =       CTIA+10
693 SIZEP3  =       CTIA+11
694 SIZEM   =       CTIA+12
695 GRAFP0  =       CTIA+13
696 GRAFP1  =       CTIA+14
697 GRAFP2  =       CTIA+15
698 GRAFP3  =       CTIA+16
699 GRAFM   =       CTIA+17
700 COLPM0  =       CTIA+18     ;PCOLR0-->COLPM0      WITH ATTRACT MODE
701 COLPM1  =       CTIA+19     ;PCOLR1-->COLPM1      WITH ATTRACT MODE
702 COLPM2  =       CTIA+20     ;PCOLR2-->COLPM2      WITH ATTRACT MODE
703 COLPM3  =       CTIA+21     ;PCOLR3-->COLPM3      WITH ATTRACT MODE
704 COLPF0  =       CTIA+22     ;COLOR0-->COLPF0      WITH ATTRACT MODE
```

```
705 COLPF1  =       CTIA+23     ;COLOR1-->COLPF1      WITH ATTRACT MODE
706 COLPF2  =       CTIA+24     ;COLOR2-->COLPF2      WITH ATTRACT MODE
707 COLPF3  =       CTIA+25     ;COLOR3-->COLPF3      WITH ATTRACT MODE
708 COLBK   =       CTIA+26     ;COLOR4-->COLBK       WITH ATTRACT MODE
709 PRIOR   =       CTIA+27     ;(ON OPEN S: OR E:)   GPRIOR-->PRIOR
710 VDELAY  =       CTIA+28
711 GRACTL  =       CTIA+29
712 HITCLR  =       CTIA+30
713 CONSOL  =       CTIA+31     ;$08-->CONSOL         TURN OFF SPEAKER
714 M0PF    =       CTIA+0
715 M1PF    =       CTIA+1
716 M2PF    =       CTIA+2
717 M3PF    =       CTIA+3
718 P0PF    =       CTIA+4
719 P1PF    =       CTIA+5
720 P2PF    =       CTIA+8
721 P3PF    =       CTIA+7
722 M0PL    =       CTIA+8
723 M1PL    =       CTIA+9
724 M2PL    =       CTIA+10
725 M3PL    =       CTIA+11
726 P0PL    =       CTIA+12
727 P1PL    =       CTIA+13
728 P2PL    =       CTIA+14
729 P3PL    =       CTIA+15
730 TRIG0   =       CTIA+16     ;TRIG0-->STRIG0
731 TRIG1   =       CTIA+17     ;TRIG1-->STRIG1
732 TRIG2   =       CTIA+18     ;TRIG2-->STRIG2
733 TRIG3   =       CTIA+19     ;TRIG3-->STRIG3
734 ;
735 ANTIC   =       $D400       ;VBLANK ACTION        DESCRIPTION
736 DMACTL  =       ANTIC+0     ;DMACTL<--SDMCTL      ON OPEN S: OR E:
737 CHACTL  =       ANTIC+1     ;CHACTL<--CHACT       ON OPEN S: OR E:
738 DLISTL  =       ANTIC+2     ;DLISTL<--SDLSTL      ON OPEN S: OR E:
739 DLISTH  =       ANTIC+3     ;DLISTH<--SDLSTH      ON OPEN S: OR E:
740 NSCROL  =       ANTIC+4
741 VSCROL  =       ANTIC+5
742 PMBASE  =       ANTIC+7
743 CHBASE  =       ANTIC+9     ;CHBASE<--CHBAS       ON OPEN S: OR E:
744 WSYNC   =       ANTIC+10
745 VCOUNT  =       ANTIC+11
746 PENH    =       ANTIC+12
747 PENV    =       ANTIC+13
748 NMIEN   =       ANTIC+14    ;NMIEN<--40 POWER     ON AND [SETVBV]
749 NMIRES  =       ANTIC+15    ;STROBED
750 NMIST   =       ANTIC+15
751 PIA     =       $D300       ;VBLANK ACTION        DESCRIPTION
752 PORTA   =       PIA+0       ;PORTA-->STICK0,1     X-Y CONTROLLERS
753 PORTB   =       PIA+1       ;PORTB-->STICK2,3     X-Y CONTROLLERS
754 PACTL   =       PIA+2       ;NONE                 PACTL<--3C [INIT]
755 PBCTL   =       PIA+3       ;NONE                 PBCTL<--3C [INIT]
756 ;
757 ;
758 ;
759 ; .PAGE
760         .PAGE
761         LIST    S
762         .TITLE  'CENTRAL INPUT/OUTPUT (CIO) 2-7-79'
763 ;               UPDATED BY AL MILLER 3-9-79
```

```
764 ASCZER  =       '0          ;ASCII ZERO
765 COLON   =       $3A         ;ASCII COLON
766 EOL     =       $9B         ;END OF RECORD
767         .PAGE
768 ;
769 ; CIO JUMP VECTOR FOR USERS
770         *=CIOV
771         JMP     CIO         ;GO TO CIO
772 ;
773 ; CIO INIT JUMP VECTOR FOR POWER UP
774         *=CIOINV
775         JMP     CIOINT      ;GO TO INIT
776 ;
777 ;
778 ; ERROR ROUTINE ADDRESS EQUATE
779 ; ERRTNH =ERRTN/256         "MOVED TO LINE 788"
780 ; ERRTNL =-ERRTNH*256+ERRTN "MOVED TO LINE 789"
781 ;
782 ;
783         *=CIOORG
784 ;
785 ; CIO INITIALIZATION (CALLED BY MONITOR AT POWER UP)
786 CIOINT: LDX     #0
787 CIOI1:  LDA     #IOCFRE     ;SET ALL IOCB'S TO FREE
788         STA     ICHID,X     ;BY SETTING HANDLER ID'S=$FF
789         LDA     #ERRTNL
790         STA     ICPTL,X     ;POINT PUT TO ERROR ROUTINE
791         LDA     #ERRTNH
792         STA     ICPTH,X
793         TXA
794         CLC
795         ADC     #IOCBSZ     ;BUMP INDEX BY SIZE
796         TAX
797         CMP     #MAXIOC     ;DONE?
798         BCC     CIOI1       ;NO
799         RTS                 ;YES, RETURN
800 ;
801 ; ERROR ROUTINE FOR ILLEGAL PUT
802 ERRTN   =*-1
803 ERRTNH  =ERRTN/256
804 ERRTNL  =(-ERRTNH)*256+ERRTN
805         LDY     #NOTOPN     ;IOCB NOT OPEN
806         RTS
807         .PAGE
808 ;
809 ; CIO LOCAL RAM (USES SPARE BYTES IN ZERO PAGE IOCB)
810 ENTVEC  =       ICSPRZ
811 ;
812 ; CIO MAIN ROUTINE
813 ;
814 ; CIO INTERFACES BETWEEN USER AND INPUT/OUTPUT DE
815 CIO:    STA     CIOCHR      ;SAVE POSSIBLE OUTPUT CHARACTER
816         STX     ICIDNO      ;SAVE IOCB NUMBER * N
817 ;
818 ; CHECK FOR LEGAL IOCB
819         TXA
820         AND     #$F         ;IS IOCB MULTIPLE OF 16?
821         BNE     CIERR1      ;NO, ERROR
822         CPX     #MAXIOC     ;IS INDEX TOO LARGE?
```

```
823         BCC     IOC1        ;NO
824 ;
825 ; INVALID IOCB NUMBER -- RETURN ERROR
826 CIERR1: LDY     #BADIOC     ;ERROR CODE
827         JMP     CIRTN1      ;RETURN
828 ;
829 ; MOVE USER IOCB TO ZERO PAGE
830 IOC1:   LDY     #0
831 IOC1A:  LDA     IOCB,X      ;USER IOCB
832         STA     IOCBAS,Y    ;TO ZERO PAGE
833         INX
834         INY
835         CPY     #12         ;12 BYTES
836         BCC     IOC1A
837 ;
838 ; COMPUTE CIO INTERNAL VECTOR FOR COMMAND
839         LDY     #NVALID     ;ASSUME INVALID CODE
840         LDA     ICCOMZ      ;COMMAND CODE TO INDEX
841         CMP     #OPEN       ;IS COMMAND LEGAL?
842         BCC     CIERR4      ;NO
843         TAY
844 ;
845 ; MOVE COMMAND TO ZERO BASE FOR INDEX
846         CPY     #SPECIL     ;IS COMMAND SPECIAL?
847         BCC     IOC2        ;NO
848         LDY     #SPECIL     ;YES, SET SPECIAL OFFSET INDEX
849 IOC2:   STY     ICCOMT      ;SAVE COMMAND FOR VECTOR
850         LDA     COMTAB-3,Y  ;GET VECTOR OFFSET FROM TABLE
851         BEQ     CIOPEN      ;GO IF OPEN COMMAND
852         CMP     #2          ;IS IT CLOSE?
853         BEQ     CICLOS      ;YES
854         CMP     #8          ;IS IT STATUS OR SPECIAL?
855         BCS     CISTSP      ;YES
856         CMP     #4          ;IS IT READ?
857         BEQ     CIREAD      ;YES
858         JMP     CIWRIT      ;ELSE, MUST BE WRITE
859         .PAGE
860 ;
861 ; OPEN COMMAND
862 ;
863 ; FIND DEVICE HANDLER IN HANDLER ADDRESS TABLE
864 CIOPEN: LDA     ICHIDZ      ;GET HANDLER ID
865         CMP     #IOCFRE     ;IS THIS IOCB CLOSED?
866         BEQ     IOC6        ;YES
867 ;
868 ; ERROR -- IOCB ALREADY OPEN
869 CIERR3: LDY     #PRVOPN     ;ERROR CODE
870 CIERR4: JMP     CIRTN1      ;RETURN
871 ;
872 ; GO FIND DEVICE
873 IOC6:   JSR     DEVSRC      ;CALL DEVICE SEARCH
874         BCS     CIERR4      ;GO IF DEVICE NOT FOUND
875 ;
876 ; DEVICE FOUND, INITIALIZE IOCB FOR OPEN
877 ;
878 ; COMPUTE HANDLER ENTRY POINT
879 IOC7:   JSR     COMENT
880         BCS     CIERR4      ;GO IF ERROR IN COMPUTE
881 ;
```

```
882 ; GO TO HANDLER FOR INITIALIZATION
883         JSR     GOHAND      ;USE INDIRECT JUMP
884 ;
885 ; STORE PUT BYTE ADDRESS-1 INTO IOCB
886         LDA     #PUTCHR     ;SIMULATE PUT CHARACTER
887         STA     ICCOMT
888         JSR     COMENT      ;COMPUTE ENTRY POINT
889         LDA     ICSPRZ      ;MOVE COMPUTED VALUE
890         STA     ICPTLZ      ;TO PUT BYTE ADDRESS
891         LDA     ICSPRZ+1
892         STA     ICPTHZ
893         JMP     CIRTN2      ;RETURN TO USER
894         .PAGE
895 ;
896 ;
897 ; CLOSE COMMAND
898 CICLOS: LDY     #SUCCES     ;ASSUME GOOD CLOSE
899         STY     ICSTAZ
900         JSR     COMENT      ;COMPUTE HANDLER ENTRY POINT
901         BCS     CICLO2      ;GO IF ERROR IN COMPUTE
902         JSR     GOHAND      ;GO TO HANDLER TO CLOSE DEVICE
903 CICLO2: LDA     #IOCFRE     ;GET IOCB "FREE" VALUE
904         STA     ICHIDZ      ;SET HANDLER ID
905         LDA     #ERRTNH
906         STA     ICPTHZ      ;SET PUT BYTE TO POINT TO ERROR
907         LDA     #ERRTNL
908         STA     ICPTLZ
909         JMP     CIRTN2      ;RETURN
910 ;
911 ;
912 ; STATUS AND SPECIAL REQUESTS
913 ; DO IMPLIED OPEN IF NECESSARY AND GO TO DEVICE
914 CISTSP: LDA     ICHIDZ      ;IS THERE A HANDLER ID?
915         CMP     #IOCFRE
916         BNE     CIST1       ;YES
917 ;
918 ; IOCB IS FREE, DO IMPLIED OPEN
919         JSR     DEVSRC      ;FIND DEVICE IN TABLE
920         BCS     CIERR4      ;GO IF ERROR IN COMPUTE
921 ;
922 ; COMPUTE AND GO TO ENTRY POINT IN HANDLER
923 CIST1:  JSR     COMENT      ;COMPUTER HANDLER ENTRY VECTOR
924         JSR     GOHAND      ;GO TO HANDLER
925 ;
926 ; RESTORE HANDLER INDEX (DO IMPLIED CLOSE)
927         LDX     ICIDNO      ;IOCB INDEX
928         LDA     ICHID,X     ;GET ORIGINAL HANDLER ID
929         STA     ICHIDZ      ;RESTORE ZERO PAGE
930         JMP     CIRTN2      ;RETURN
931         .PAGE
932 ;
933 ; READ -- DO GET COMMANDS
934 CIREAD: LDA     ICCOMZ      ;GET COMMAND BYTE
935         AND     ICAX1Z      ;IS THIS READ LEGAL?
936         BNE     RCI1A       ;YES
937 ;
938 ; ILLEGAL READ -- IOCB OPENED FOR WRITE ONLY
939         LDY     #WRONLY     ;ERROR CODE
940 RCI1B:  JMP     CIRTN1      ;RETURN
```

```
941 ;
942 ; COMPUTE AND CHECK ENTRY POINT
943 RCI1A:  JSR     COMENT      ;COMPUTE ENTRY POINT
944         BCS     RCI1B       ;GO IF ERROR IN COMPUTE
945 ;
946 ; GET RECORD OR CHARACTERS
947         LDA     ICBLLZ
948         ORA     ICBLLZ+1    ;IS BUFFER LENGTH ZERO?
949         BNE     RCI3        ;NO
950         JSR     GOHAND
951         STA     CIOCHR
952         JMP     CIRTN2
953 ;
954 ; LOOP TO FILL BUFFER OR END RECORD
955 RCI3:   JSR     GOHAND      ;GO TO HANDLER TO GET BYTE
956         STA     CIOCHR      ;SAVE BYTE
957         BMI     RCI4        ;END TRANSFER IF ERROR
958         LDY     #0
959         STA     (ICBALZ),Y  ;PUT BYTE IN USER BUFFER
960         JSR     INCBFP      ;INCREMENT BUFFER POINTER
961         LDA     ICCOMZ      ;GET COMMAND CODE
962         AND     #2          ;IS IT GET RECORD?
963         BNE     RCI1        ;NO
964 ;
965 ; CHECK FOR EOL ON TEXT RECORDS
966         LDA     CIOCHR      ;GET BYTE
967         CMP     #EOL        ;IS IT AN EOL?
968         BNE     RCI1        ;NO
969         JSR     DECBFL      ;YES, DECREMENT BUFFER LENGTH
970         JMP     RCI4        ;END TRANSFER
971 ;
972 ; CHECK BUFFER FULL
973 RCI1:   JSR     DECBFL      ;DECREMENT BUFFER LENGTH
974         BNE     RCI3        ;CONTINUE IF NON ZERO
975         .PAGE
976 ;
977 ; BUFFER FULL. RECORD NOT ENDED
978 ; DISCARD BYTES UNTIL END OF RECORD
979 RCI2:   LDA     ICCOMZ      ;GET COMMAND BYTE
980         AND     #2          ;IS IT GET CHARACTER?
981         BNE     RCI4        ;YES, END TRANSFER
982 ;
983 ; LOOP TO WAIT FOR EOL
984 RCI6:   JSR     GOHAND      ;GET BYTE FROM HANDLER
985         STA     CIOCHR      ;SAVE CHARACtER
986         BMI     RCI4        ;GO IF ERROR
987 ;
988 ; TEXT RECORD. WAIT FOR EOL
989         LDA     CIOCHR      ;GET GOT BYTE
990         CMP     #EOL        ;IS IT EOL?
991         BNE     RCI6        ;NO, CONTINUE
992 ;
993 ; END OF RECORD. BUFFER FULL -- SEND TRUNCATED RECORD MESSAGE
994 RCI1I:  LDA     #TRNRCD     ;ERROR CODE
995         STA     ICSTAZ      ;STORE IN 10GB
996 ;
997 ; TRANSFER DONE
998 RCI4:   JSR     SUBBFL      ;SET FINAL BUFFER LENGTH
999         JMP     CIRTN2      ;RETURN
```

```
1000         .PAGE
1001 ;
1002 ; WRITE -- DO PUT COMMANDS
1003 CIWRIT: LDA     ICCOMZ      ;GET COMMAND BYTE
1004         AND     ICAX1Z      ;IS THIS WRITE LEGAL?
1005         BNE     WCIIA       ;YES
1006 ;
1007 ; ILLEGAL WRITE -- DEVICE OPENED FOR READ ONLY
1008         LDY     #RDONLY     ;ERROR CODE
1009 WCI1B:  JMP     CIRTN1      ;RETURN
1010 ;
1011 ;  COMPUTE AND CHECK ENTRY POINT
1012 WCIIA:  JSR     COMENT      ;COMPUTE HANDLER ENTRY POINT
1013         BCS     WCI1B       ;GO IF ERROR IN COMPUTE
1014 ;
1015 ; PUT RECORD OR CHARACTERS
1016         LDA     ICBLLZ
1017         ORA     ICBLLZ+1    ;IS BUFFER LENGTH ZERO?
1018         BNE     WCI3        ;NO
1019         LDA     CIOCHR      ;GET CHARACTER
1020         INC     ICBLLZ      ;SET SUFFER LENOTHI
1021         BNE     WCI4        ;THEN JUST TRANSFER ONE BYTE
1022 ;
1023 ; LOOP TO TRANSFER BYTES FROM BUFFER TO HANDLER
1024 WCI3:   LDY     #0
1025         LDA     (ICBALZ),Y  ;GET BYTE FROM BUFFER
1026         STA     CIOCHR      ;SAVE
1027 WCI4:   JSR     GOHAND      ;GO PUT BYTE
1028         BMI     WCI5        ;END IF ERROR
1029         JSR     INCBFP      ;INCREMENT BUFFER POINTER
1030 ;
1031 ; CHECK FOR TEXT RECORD
1032         LDA     ICCOMZ      ;GET COMMAND BYTE
1033         AND     #2          ;IS IT PUT RECORD?
1034         BNE     WCI1        ;NO
1035 ;
1036 ; TEXT RECORD -- CHECK FOR EOL TRANSFER
1037         LDA     CIOCHR      ;GET LAST CHARACTER
1038         CMP     #EOL        ;IS IT AN EOL?
1039         BNE     WCI1        ;NO
1040         JSR     DECBFL      ;DECREMENT BUFFER LENGTH
1041         JMP     WCI5        ;END TRANSFER
1042 ;
1043 ; CHECK FOR BUFFER EMPTY
1044 WCI1:   JSR     DECBFL      ;DECREMENT BUFFER LENGTH
1045         BNE     WCI3        ;CONTINUE IF NON ZERO
1046         .PAGE
1047 ;
1048 ; BUFFER EMPTY, RECORD NOT FILLED
1049 ; CHECK TYPE OF TRANSFER
1050 WCI2:   LDA     ICCOMZ      ;GET COMMAND CODE
1051         AND     #2          ;IS IT PUT CHARACTER?
1052         BNE     WCI5        ;YES, END TRANSFER
1053 ;
1054 ; PUT RECORD (TEXT), BUFFER ,EMPTY, SEND EOL
1055         LDA     #EOL
1056         JSR     GOHAND      ;GO TO HANDLER
1057 ;
1058 ; END PUT TRANSFER
```

```
1059 WCI5:   JSR     SUBBFL      ;SET ACTUAL PUT BUFFER LENGTH
1060         JMP     CIRTN2      ;RETURN
1061         .PAGE
1062 ;
1063 ; CIO RETURNS
1064 ; RETURNS WITH Y=STATUS
1065 CIRTN1: STY     ICSTAZ      ;SAVE STATUS
1066 ;
1067 ; RETURNS WITH STATUS STORED IN ICSTAZ
1068 ; MOVE IOCB IN ZERO PAGE BACK TO USER AREA
1069 CIRTN2: LDY     ICIDNO      ;GET IOCB INDEX
1070         LDA     ICBAL,Y
1071         STA     ICBALZ      ;RESTORE USER BUFFER POINTER
1072         LDA     ICBAH,Y
1073         STA     ICBAHZ
1074         LDX     #0          ;LOOP COUNT AND INDEX
1075 CIRT3:  LDA     IOCBAS,X    ;ZERO PAGE
1076         STA     IOCB,Y      ;TO USER AREA
1077         INX
1078         INY
1079         CPX     #12         ;12 BYTES
1080         BCC     CIRT3
1081 ;
1082 ; RESTORE A,X, & Y
1083         LDA     CIOCHR      ;GET LAST CHARACTER
1084         LDX     ICIDNO      ;IOCB INDEX
1085         LDY     ICSTAZ      ;GET STATUS AND SET FLAGS
1086         RTS                 ;RETURN TO USER
1087         .PAGE
1088 ;
1089 ;
1090 ; CIO SUBROUTINES
1091 ;
1092 ; COMENT -- CHECK AND COMPUTE HANDLER ENTRY POINT
1093 COMENT: LDY     ICHIDZ      ;GET HANDLER INDEX
1094         CPY     #MAXDEV+1   ;IS IT A LEGAL INDEX?
1095         BCC     COM1        ;YES
1096 ;
1097 ; ILLEGAL HANDLER INDEX MEANS DEVICE NOT OPEN FOR OPERATION
1098         LDY     #NOTOPN     ;ERROR CODE
1099         BCS     COM2        ;RETURN
1100 ;
1101 ; USE HANDLER ADDRESS TABLE AND COMMAND TABLE TO GET VECTOR
1102 COM1:   LDA     HATABS+1,Y  ;GET LOW BYTE OF ADDRESS
1103         STA     ICSPRZ      ;AND SAVE IN POINTER
1104         LDA     HATABS+2,Y  ;GET HI BYTE OF ADDRESS
1105         STA     ICSPRZ+1
1106         LDY     ICCOMT      ;GET COMMAND CODE
1107         LDA     COMTAB-3,Y  ;GET COMMAND OFFSET
1108         TAY
1109         LDA     (ICSPRZ),Y  ;GET LOW BYTE OF VECTOR FROM
1110         TAX                 ;HANDLER ITSELF AND SAVE
1111         INY
1112         LDA     (ICSPRZ),Y  ;GET HI BYTE OF VECTOR
1113         STA     ICSPRZ+1
1114         STX     ICSPRZ      ;SET LO BYTE
1115         CLC                 ;SHOW NO ERROR
1116 COM2:   RTS
1117 ;
```

```
1118 ;
1119 ; DECBFL -- DECREMENT BUFFER LENGTH DOUBLE BYTE
1120 ; Z FLAG = 0 ON RETURN IF LENGTH = 0 AFTER DECREMENT
1121 DECBFL: DEC     ICBLLZ      ;DECREMENT LOW BYTE
1122         LDA     ICBLLZ      ;CHECK IT
1123         CMP     #$FF        ;DID IT GO BELOW?
1124         BNE     DECBF1      ;NO
1125         DEC     ICBLLZ+1    ;DECREMENT HI BYTE
1126 DECBF1: ORA     ICBLLZ+1    ;SET Z IF BOTH ARE ZERO
1127         RTS
1128 ;
1129 ;
1130 ; INCBFP -- INCREMENT WORKING BUFFER POINTER
1131 INCBFP: INC     ICBALZ      ;BUMP LOW BYTE
1132         BNE     INCBF1      ;GO IF NOT ZERO
1133         INC     ICBALZ+1    ;ELSE, BUMP HI BYTE
1134 INCBF1: RTS
1135 ;
1136 ;
1137 ; SUBBFL -- SET BUFFER LENGTH = BUFFER LENGTH - WORKING BYTE COUNT
1138 SUBBFL: LDX     ICIDNO      ;GET IOCB INDEX
1139         SEC
1140         LDA     ICBLL,X     ;GET LOW BYTE OF INITIAL LENGTH
1141         SBC     ICBLLZ       ;SUBTRACT FINAL LOW BYTE
1142         STA     ICBLLZ      ;AND SAVE BACK
1143         LDA     ICBLH,X     ;GET HI BYTE
1144         SBC     ICBLLZ+1
1145         STA     ICBLHZ
1146         RTS
1147 ;
1148 ;
1149 ; GOHAND -- GO INDIRECT TO A DEVICE HANDLER
1150 ; Y= STATUS ON RETURN, N FLAG=1 IF ERROR ON RETURN
1151 GOHAND: LDY     #FNCNOT     ;PREPARE NO FUNCTION STATUS FOR HANDLER RTS
1152         JSR     CIJUMP      ;USE THE INDIRECT JUMP
1153         STY     ICSTAZ      ;SAVE STATUS
1154         CPY     #0          ;AND SET N FLAG
1155         RTS
1156 ;
1157 ; INDIRECT JUMP TO HANDLER BY PAUL'S METHOD
1158 CIJUMP: TAX                 ;SAVE A
1159         LDA     ICSPRZ+1    ;GET JUMP ADDRESS HI BYTE
1160         PHA                 ;PUT ON STACK
1161         LDA     ICSPRZ      ;GET JUMP ADDRESS LO BYTE
1162         PHA                 ;PUT ON STACK
1163         TXA                 ;RESTORE A
1164         LDX     ICIDNO      ;GET IOCB INDEX
1165         RTS                 ;GO TO HANDLER INDIRECTLY
1166         .PAGE
1167 ;
1168 ; DEVSRC -- DEVICE SEARCH, FIND DEVICE IN HANDLER ADDRESS TABLE
1169 ;
1170 ; LOOP TO FIND DEVICE
1171 DEVSRC: LDY     #0
1172         LDA     (ICBALZ),Y  ;GET DEVICE NAME FROM USER
1173         BEQ     CIERR2
1174         LDY     #MAXDEV     ;INITIAL COMPARE INDEX
1175 DEVS1:  CMP     HATABS,Y    ;IS THIS THE DEVICE?
1176         BEQ     DEVS2       ;YES
```

```
1177         DEY
1178         DEY                 ;ELSE, POINT TO NEXT DEVICE NAME
1179         DEY
1180         BPL     DEVS1       ;CONTINUE FOR ALL DEVICES
1181 ;
1182 ; NO DEVICE FOUND, DECLARE NON-EXISTENT DEVICE ERROR
1183 CIERR2: LDY     #NONDEV     ;ERROR CODE
1184         SEC                 ;SHOW ERROR
1185         BCS     DEVS4       ;AND RETURN
1186 ;
1187 ; FOUND DEVICE, SET ICHID,ICDNO, AND INIT DEVICE
1188 DEVS2:  TYA
1189         STA     ICHIDZ      ;SAVE HANDLER INDEX
1190         SEC
1191         LDY     #1
1192         LDA     (ICBALZ),Y  ;GET DEVICE NUMBER (DRIVE NUMBER)
1193         SBC     #ASCZER     ;SUBTRACT ASCII ZERO
1194         CMP     #$A         ;IS NUMBER IN RANGE?
1195         BCC     DEVS3       ;YES
1196         LDA     #1          ;NO. DEFAULT TO ONE
1197 DEVS3:  STA     ICDNOZ      ;SAVE DEVICE NUMBER
1198         CLC                 ;SHOW NO ERROR
1199 ;
1200 ; RETURN
1201 DEVS4:  RTS
1202         .PAGE
1203 ;
1204 ;
1205 ; CIO ROM TABLES
1206 ;
1207 ; COMMAND TABLE
1208 ; MAPS EACH COMMAND TO OFFSET FOR APPROPRIATE VECTOR IN HANDLER
1209 COMTAB: BYTE    0,4,4,4,4,6,6,6,6,2,8,10
1210
1211
1212 LENGTH  =*-CIOINT
1213 CRNTP1  =*
1214         *=$14
1215 CIOSPR: BYTE    INTORG-CRNTP1 ;^GCIOL IS TOO LONG
1216 ;
1217         .TITLE  'INTERRUPT HANDLER'
1218 ;LIVES ON DK1:INTHV.SRC
1219 SRTIM2  =       6           ;SECOND REPEAT INTERVAL
1220 ;
1221 ; THIS IS TO MAKE DOS 2 WORK WHICH USED AN ABSOLUTE ADDRESS
1222 ;
1223         *=$E912
1224         JMP     SETVBL
1225         *=SETVBV
1226         JMP     SETVBL
1227         JMP     SYSVBL
1228         JMP     XITVBL
1229         *=INTINV
1230         JMP     IHINIT
1231 ;
1232         *=VCTABL+INTABS-VDSLST
1233 ;
1234         .WORD   SYRTI       ;VDSLST
1235         .WORD   SYIRQB      ;VPRCED
```

```
1236         .WORD   SYIRQB      ;VINTER
1237         .WORD   SYIRQB      ;VBREAK
1238 ;
1239         .RES    8
1240         .WORD   SYIRQB      ;VTMIR1
1241         .WORD   SYIRQB      ;VTIMR2
1242         .WORD   SYIRQB      ;VTMIR4
1243         .WORD   SYIRQ       ;VIMIRQ
1244         .WORD   0,0,0,0,0   ;CDTMV1-4
1245
1246
1247         .WORD   SYSVBL      ;VVBLKI
1248         .WORD   XITVBL      ;VVSLKD
1249 ;
1250         *=$900C
1251 ;
1252         LDA     #PIRQH      ;SET UP RAM VECTORS FOR LINBUG VERSION
1253         STA     $FFF9
1254         LDA     #PIRQL
1255         STA     $FFF8
1256         LDA     #PNMIH
1257         STA     $FFFB
1258         LDA     #PNMIL
1259         STA     $FFFA
1260         RTS
1261         .PAGE
1262 ;
1263 ; IRQ HANDLER
1264 ;
1265 ; JUMP THRU IMMEDIATE IRQ VECTOR, WHICH ORDINARILY POINTS TO
1266 ; SYSTEM IRQ; DETERMINE & CLEAR CAUSE, JUMP THRU SOFTWARE VECTOR.
1267 ;
1268         *=INTORG
1269 IHINIT: LDA     #$40        ;VBL ON BUF DLIST OFF***FOR NOW***
1270         STA     NMIEN       ;ENABLE DISPLAY LIST, VERTICAL BLANK
1271         LDA     #$38        ;LOOK AT DATA DIRECTION REGISTERS IN PIA
1272         STA     PACTL
1273         STA     PBCTL
1274         LDA     #0          ;MAKE ALL INPUTS
1275         STA     PORTA
1276         STA     PORTB
1277         LDA     #$3C        ;BACK TO PORTS
1278         STA     PACTL
1279         STA     PBCTL
1280         RTS
1281 PIRQ:   JMP     (VIMIRU)
1282 CMPTAB: .BYTE   $80         ;BREAK KEY
1283         .BYTE   $40         ;KEY STROKE
1284         .BYTE   $04         ;TIMER 4
1285         .BYTE   $02         ;TIMER 2
1286         .BYTE   $01         ;TIMER 1
1287         .BYTE   $08         ;SERIAL OUT COMPLETE
1288         .BYTE   $10         ;SERIAL OUT READY
1289         .BYTE   $20         ;SERIAL IN READY
1290
1291 ; THIS IS A TABLE OF OFFSETS INTO PAGE 2.  THEY POINT TO
1292 ADRTAB: .BYTE   BRKKY-INTABS
1293         .BYTE   VKEYBD-INTABS
1294         .BYTE   VTIMR4-INTABS
```

```
1295         .BYTE   VTIMR2-INTABS
1296         .BYTE   VTIMR1-INTABS
1297         .BYTE   VSEROC-INTABS
1298         .BYTE   VSEROR-INTABS
1299         .BYTE   VSERIN-INTABS
1300
1301 SYIRQ:  PHA                    ;SAVE ACCUMULATOR
1302         LDA     IRQST          ; CHECK FOR SERIAL IN
1303         AND     #$20
1304         BNE     SYIRQ2
1305         LDA     #$DF           ; MASK ALL OTHERS
1306         STA     IRQEN
1307         LDA     POKMSK
1308         STA     IRQEN
1309         JMP     (VSERIN)
1310 SYIRQ2: TXA                    ;PUT X INTO ACC
1311         PHA                    ;SAVE K ONTO STACK
1312         LDX     #$6            ;START WITH SIX OFFSET
1313 LOOPM:  LDA     CMPTAB,X       ;LOAD MASK
1314         CPX     #5             ;CHECK TO SEE IF COMPLETE IS SET
1315         BNE     LOOPM2
1316         AND     POKMSK         ;IS THIS INTERUPT ENABLED?
1317         BEQ     LL
1318 LOOPM2: BIT     IRQST          ; IS IT THE INTERUPT?
1319         BEQ     JMPP
1320 LL:     DEX                    ;NO DEC X AND TRY NEXT MASK
1321         BPL     LOOPM          ;IF NOT NEC 0010 LOOPH
1322         JMP     SYIRQ8         ;DONE BUT NO INTERUPT
1323 JMPP:   EOR     #$FF           ;COMPLEMENT MASK
1324         STA     IRQEN          ;ENABLE ALL OTHERS
1325         LDA     POKMSK         ; GET POKE MASK
1326         STA     IRQEN          ; ENABLE THOSE IN POKE MASK
1327         LDA     ADRTAB,X
1328         TAX
1329         LDA     INTABS,X       ; GET ADDRESS LOW PART
1330         STA     JVECK          ; PUT IN VECTOR
1331         LDA     INTABS+1,X     ; GET ADDRESS HIGH PART
1332         STA     JVECK+1        ; PUT IN VECTOR HIGH PART
1333         PLA                    ; PULL X REGISTER FROM STACK
1334         TAX                    ; PUT IT INTO X
1335         JMP     (JVECK)        ; JUMP TO THE PROPER ROUTINE
1336 BRKKY2: LDA     #0             ; BREAK KEY ROUTINE
1337         STA     BRKKEY         ; SET BREAK KEY FLAG
1338         STA     SSFLAG         ; START/STOP FLAG
1339         STA     CRSINH         ; CURSOR INHIBIT
1340         STA     ATRACT         ; TURN OFF ATRACT MODE
1341         PLA
1342         RTI                    ;EXIT FROM INT
1343 SYIRQ8: PLA
1344         TAX
1345         BIT     PACTL          ;PROCEED ***I GUESS***
1346         BPL     SYIRQ9
1347         LDA     PORTA          ;CLEAR INT STATUS BIT
1348         JMP     (VPRCED)
1349 SYIRQ9: BIT     PBCTL          ;INTERRUPT ***I GUESS***
1350         BPL     SYIRQA
1351         LDA     PORTB          ;CLEAR INT STATUS
1352         JMP     (VINTER)
1353 SYIRQA: PLA
```

```
1354         STA     JVECK
1355         PLA
1356         PHA
1357         AND     #$10        ;B BIT OF P REGISTER
1358         BEQ     SYRTI2
1359         LDA     JVECK
1360         PHA
1361         JMP     (VBREAK)
1362 SYRTI2: LDA     JVECK
1363         PHA
1364 SYIRQB: PLA
1365 SYRTI:  RTI                 ;UNIDENTIFIED INTERRUPT, JUST RETURN
1366         .PAGE
1367 ;
1368 ; NMI HANDLER
1369 ;
1370 ; DETERMINE CAUSE AND JUMP THRU VECTOR
1371 ;
1372 PNMI:   BIT     NMIST
1373         BPL     PNMI1       ;SEE IF DISPLAY LIST
1374         JMP     (VDSLST)
1375 PNMI1:  PHA
1376         LDA     NMIST
1377         AND     #$20        ;SEE IF RESET
1378         BEQ     *+5
1379         JMP     WARMSV      ;DO THRU WARM START JUMP
1380         TXA                 ;SAVE REGISTERS
1381         PHA
1382         TYA
1383         PHA
1384         STA     NMIRES      ;RESET INTERRUPT STATUS
1385         JMP     (VVBLKI)    ;JUMP THRU VECTOR
1386         .PAGE
1387 ;
1388 ; SYSTEM VBLANK ROUTINE
1389 ;
1390 ; INC FRAME COUNTER. PROCESS COUNTDOWN TIMERS. EXIT IF I WAS SET. CLEAR
1391 ; SET DLISTL, DLISTH, DMACTL FROM RAM CELLS. DO SOFTWARE REPEAT.
1392 ;
1393 SYSVBL: INC     RTCLOK+2    ;INC FRAME COUNTER
1394         BNE     SYSVB1
1395         INC     ATRACT      ;INCREMENT ATRACT (CAUSES ATRACT WHEN MINUS)
1396         INC     RTCLOK+1
1397         BNE     SYSVB1
1398         INC     RTCLOK
1399 SYSVB1: LDA     #$FE        ;{ATRACT] SET DARK MASK TO NORMAL
1400         LDX     #0          ;SET COLRSH TO NORMAL
1401         LDY     ATRACT      ;TEST ATRACT FOR NEGATIVE
1402         BPL     VBATRA      ;WHILE POSITIVE DONT GO INTO ATRACT
1403         STA     ATRACT      ;IN ATRACTI SO STAY BY STA $FE
1404         LDX     RTCLOK+1    ;COLOR SHIFT FOLLOWS RICLOK+1
1405         LDA     #$F6        ;SET DARK MASK TO DARK
1406 VBATRA: STA     DRKMSK
1407         STX     COLRSH
1408         LDX     #0          ;POINT TO TIMER1
1409         JSR     DCTIMR      ;GO DECREMENT TIMER1
1410         BNE     SYSVB2      ;BRANCH IF STILL COUNTING
1411         JSR     JTIMR1      ;GO JUMP TO ROUTINE
1412 SYSVB2: LDA     CRITIC
```

```
1413         BNE     XXIT         ;GO IF CRITICAL SET
1414         TSX                  ;SEE IF I WAS SET
1415         LDA     $104,X       ;GET STACKED P
1416         AND     #$04         ;I BIT
1417         BEQ     SYSVB3       ;BRANCH IF OK
1418 XXIT:   JMP     XITVBL       ;I WAS SET, EXIT
1419 SYSVB3: LDA     PENV
1420         STA     LPENV
1421         LDA     PENH
1422         STA     LPENH
1423         LDA     SDLSTH
1424         STA     DLISTH
1425         LDA     SDLSTL
1426         STA     DLISTL
1427         LDA     SDMCTL
1428         STA     DMACTL
1429         LDA     GPRIOR       ;GLOBAL PRIOR
1430         STA     PRIOR
1431         LDX     #$08         ;TURN OFF KEYBOARD SPEAKER
1432         STX     CONSOL
1433 SCOLLP: CLI                  ;DISABLE INTERUPTS
1434         LDA     PCOLR0, X    ;LOAD COLOR REGISTERS FROMRAM
1435         EOR     COLRSH       ;DO COLOR SHIFT
1436         AND     DRKMSK       ;AND DARK ATRACT
1437         STA     COLPM0,X
1438         DEX
1439         BPL     SCOLLP
1440         LDA     CHBAS
1441         STA     CHBASE
1442         LDA     CHACT
1443         STA     CHACTL
1444         LDX     #2           ;POINT TO TIMER 2
1445         JSR     DCTIMR
1446         BNE     SYSVB4       ;IF DIDNT GO ZERO
1447         JSR     JTIMR2       ;GO JUMP TO TIMER2 ROUTINE
1448 SYSVB4: LDX     #2           ;RESTORE X
1449 SYSVBB: INX
1450         INX
1451         LDA     CDTMV1,X
1452         ORA     CDTMV1+1,X
1453         BEQ     SYSVBA
1454         JSR     DCTIMR       ;DECREMENT AND SET FLAG IF NONZERO
1455         STA     CDTMF3-4,X
1456 SYSVBA: CPX     #8           ;SEE IF DONE ALL 3
1457         BNE     SYSVBB       ;LOOP
1458 ; CHECK DEBOUNCE COUNTER
1459         LDA     SKSTAT
1460         AND     #$04         ;KEY DOWN BIT
1461         BEQ     SYVB6A       ;IF KEY DOWN
1462 ; KEY UP SO COUNT IT
1463         LDA     KEYDEL       ;KEY DELAY COUNTER
1464         BEQ     SYVB6A       ;IF COUNTED DOWN ALREADY
1465         DEC     KEYDEL       ;COUNT IT
1466 ; CHECK SOFTWARE REPEAT TIMER
1467 SYVB6A: LDA     SRTIMR
1468         BEQ     SYSVB7       ;DOESN'T COUNT
1469         LDA     SKSTAT
1470         AND     #$04         ;CHECK KEY DOWN BIT
1471         BNE     SYSVB6       ;BRANCH IF NO LONGER DOWN
```

```
1472         DEC     SRTIMR      ;COUNT FRAME OF KEY DOWN
1473         BNE     SYSVB7      ;BRANCH IF NOT RUN OUT
1474 ; TIMER RAN OUT - RESET AND SIMULATE KEYBOARD IRQ
1475         LDA     #SRTIM2     ;TIMER VALUE
1476         STA     SRTIMR      ;SET TIMER
1477         LDA     KBCODE      ;GET THE KEY
1478         STA     CH          ;PUT INTO CH
1479 ; READ GAME CONTROLLERS
1480 SYSVB7: LDY     #1
1481         LDX     #3
1482 STLOOP: LDA     PORTA,Y
1483         LSR     A
1484         LSR     A
1485         LSR     A
1486         LSR     A
1487         STA     STICK0,X    ;STORE JOYSTICK
1488         DEX
1489         LDA     PORTA,Y
1490         AND     #$F
1491         STA     STICK0,X    ;STORE JOYSTICK
1492         DEX
1493         DEY
1494         BPL     STLOOP
1495 ;
1496         LDX     #3
1497 STRL:   LDA     TRIG0,X     ;MOVE JOYSTICK TRIGGERS
1498         STA     STRIG0,X
1499         LDA     POT0,X      ;MOVE POT VALUES
1500         STA     PADDL0,X
1501         LDA     POT4,X
1502         STA     PADDL4,X
1503         DEX
1504         BPL     STRL
1505         STA     POTGO       ;START POTS FOR NEXT TIME
1506 ;
1507         LDX     #6
1508         LDY     #3
1509 PTRLP:  LDA     STICK0,Y    ;TRANSFER BITS FROM JOYSTICKS
1510         LSR     A           ;TO PADDLE TRIGGERS
1511         LSR     A
1512         LSR     A
1513         STA     PTRIG1,X
1514         LDA     #0
1515         ROL     A
1516         STA     PTRIG0,X
1517         DEX
1518         DEX
1519         DEY
1520         BPL     PTRLP
1521 ;
1522         JMP     (VVBLKD)    ;GO TO DEFERRED VBLANK ROUTINE
1523 SV7H    =       SYSVB7/256
1524 SV7L    =       (-256)*SV7H+SYSVB7
1525 SYSVB6: LDA     #0
1526         STA     SRTIMR      ;ZERO TIMER
1527         BEQ     SYSVB7      ;UNCOND
1528 JTIMR1: JMP     (CDTMA1)
1529 JTIMR2: JMP     (CDTMA2)
1530 ;
```

```
1531 ; SUBROUTINE TO DECREMENT A COUNTDOWN TIMER
1532 ; ENTRY X=OFFSET FROM TIMER 1
1533 ; EXIT A,P=ZERO IF WENT ZERO, FF OTHERWISE
1534 ;
1535 DCTIMR: LDY     CDTMV1,X    ;LO BYTE
1536         BNE     DCTIM1      ;NONZERO, GO DEC IT
1537         LDY     CDTMV1+1,X  ;SEE IF BOTH ZERO
1538         BEQ     DCTXF       ;YES, EXIT NONZERO
1539         DEC     CDTMV1+1,X  ;DEC HI BYTE
1540 DCTIM1: DEC     CDTMV1,X    ;DEC LO BYTE
1541         BNE     DCTXF
1542         LDY     CDTMV1+1,X
1543         BNE     DCTXF
1544         LDA     #0          ;WENT ZERO. RETURN ZERO
1545         RTS
1546 DCTXF:  LDA     #$FF        ;RETURN NONZERO
1547         RTS
1548         .PAGE
1549 ;
1550 ; SUBROUTINE TO SET VERTICAL BLANK VECTORS AND TIMERS
1551 ; ENTRY X=HI,Y=LO BYTE TO SET
1552 ;       A= 1-5 TIMERS 1-5
1553 ;          6 IMM VBLANK
1554 ;          7 DEF VBLANK
1555 ;
1556 SETVBL: ASL     A           ;MUL BY 2
1557         STA     INTEMP
1558         TXA
1559         LDX     #5
1560         STA     WSYNC       ;WASTE 20 CPU CYCLES
1561 SETLOP: DEX                 ;TO ALOWD VBLANK TO HAPPEN
1562         BNE     SETLOP      ;IF THIS IS LINE "7C"
1563         LDX     INTEMP
1564         STA     CDTMV1-1,X
1565         TYA
1566         STA     CDTMV1-2,X
1567         RTS
1568 ;
1569 ; EXIT FROM VERTICAL BLANK
1570 ;
1571 XITVBL: PLA                 ;UNSTACK Y
1572         TAY
1573         PLA                 ;UNSTACK X
1574         TAX
1575         PLA                 ;UNSTACK A
1576         RTI                 ;AND GO BACK FROM WHENCE.
1577 PIRQH   =       PIRQ/256
1578 PIRQL   =       (-256)*PIRQH+PIRQ
1579 PNMIH   =       PNMI/256
1580 PNMIL   =       (-256)*PNMIH+PNMI
1581 ; SPARE BYTE OR MODULE TOO LONG FLAG
1582 CRNTP2  =*
1583         *=$14
1584 INTSPR: .BYTE   SIOORG-CRNTP2 ;^GINTHV IS TOO LONG
1585 ;
1586         .TITLE 'SIO ( SERIAL BUS INPUT/OUTPUT CONTROLLER )'
1587 ;       COLLEEN OPERATING SYSTEM
1588 ;
1589 ;       SIO  ( SERIAL BUS INPUT/OUTPUT CONTROLLER )
```

```
1590 ;       WITH SOFTWARE BAUD RATE CORRECTION ON CASSETTE
1591 ;
1592 ;
1593 ;               AL MILLER       3-APR-19
1594 ;
1595 ;
1596 ; THIS MODULE HAS ONE ENTRY POINT.  IT IS CALLED BY THE DEVICE
1597 ; HANDLERS.  IT INTERPRETS A PREVIOUSLY ESTABLISHED DEVICE CONTROL
1598 ; BLOCK (STORED IN GLOBAL RAM) TO ISSUE COMMANDS
1599 ; TO THE SERIAL BUS TO CONTROL TRANSMITTING AND RECEIVING DATA.
1600 ;
1601 ;
1602 ;
1603 ;
1604         .PAGE
1605 ; EQUATES
1606 ;
1607 ; DCD DEVICE BUS ID NUMBERS
1608 FLOPPY  =       $30
1609 ;PRINTR =       $40
1610 ;CASSET =       $80         ;!!!!! *****
1611 CASET   =       $60         ;!!!!! *****
1612 ;
1613 ;
1614 ;BUS COMMANDS
1615 ;
1616 READ    =       'R
1617 WRITE   =       'W
1618 ;STATIS = 'S
1619 ;FORMAT = '!
1620 ;
1621 ;
1622 ; COMMAND AUX BYTES
1623 ;
1624 SIDWAY  =       'S          ;PRINT 18 CHARACTERS SIDEWAYS
1625 NORMAL  =       'N          ;PRINT 40 CHARACTERS NORMALLY
1626 DOUBLE  =       'D          ;PRINT 20 CHARACTERS DOUBLE WIDE
1627 PLOT    =       'P          ;PLOT MODE
1628 ;
1629 ;
1630 ; BUS RESPONSES
1631 ;
1632 ACK     =       'A          ;DEVICE ACKNOWLEDGES INFORMATION
1633 NACK    =       'N          ;DEVICE DID NOT UNDERSTAND
1634 COMPLT  =       'C          ;DEVICE SUCCESSFULLY COMPLETED OPERATION
1635 ERROR   =       'E          ;DEVICE INCURRED AN ERROR IN AN ATTEMPTED OP
1636 ;
1637 ;
1638 ; MISCELLANEOUS EQUATES
1639 ;
1640 B192LO  =       $28         ;19200 BAUD RATE POKEY COUNTER VALUES (LO BY
1641 B192HI  =       $00         ;(HI BYTE)
1642 B600LO  =       $CC         ;600 BAUD (LO BYTE)
1643 B600HI  =       $05         ;(HI BYTE)
1644 HITONE  =       $05         ;FSK HI FREQ POKEY COUNTER VALUE (5326 HZ)
1645 LOTONE  =       $07         ;FSK LO FREQ POKEY COUNTER VALUE (3995 HZ)
1646 ;
1647         .IF     PALFLG
1648 WIRGLO  =       150         ;WRITE INTER RECORD GAP (IN 1/60 SEC)
```

```
1649 RIRGLO  =       100         ;READ INTER RECORD GAP (IN 1/60 SEC)
1650 WSIRG   =        13         ;SHORT WRITE INTER RECORD GAP
1651 RSIRG   =         8         ;SHORT READ INTER RECORD GAP
1652         .ENDIF
1653         .IF     PALFLG-1
1654 WIRGLO  =       180         ;WRITE INTER RECORD GAP (IN 1/60 SEC)
1655 RIRGLO  =       120         ;READ INTER RECORD GAP (IN 1/60 SEC)
1656 WSIRG   =       15          ;SHORT WRITE INTER RECORD GAP
1657 RSIRG   =       10          ;SHORT READ INTER RECORD GAP
1658         .ENDIF
1659 WIRGHI  =       0
1660 RIRGHI  =       0
1661 ;
1662 NCOMLO  =       $34         ;PIA COMMAND TO LOWER NOT COMMAND LINE
1663 NCOMHI  =       $3C         ;PIA COMMAND TO RAISE NOT COMMAND LINE
1664 MOTRGO  =       $34         ;PIA COMMAND TO TURN ON CASSETTE MOTOR
1665 MOTRST  =       $3C         ;PIA COMMAND TO TURN OFF MOTOR
1666 ;
1667 TEMPHI  =       TEMP/256    ;ADDRESS OF TEMP CELL (HI BYTE)
1668 TEMPLO  =       (-256)*TEMPHI+TEMP ;(LO BYTE)
1669 CBUFHI  =       CDEVIC/256  ;ADDRESS OF COMMAND BUFFER (HI BYTE)
1670 CBUFLO  =       (-256)*CBUFHI+CDEVIC ;(LO BYTE)
1671 ;
1672 CRETRI  =       13          ;NUMBER OF COMMAND FRAME RETRIES
1673 DRETRI  =       1           ;NUMBER OF DEVICE RETRIES
1674 CTIMLO  =       2           ;COMMAND FRAME ACK TIME OUT (LO BYTE)
1675 CTIMHI  =       0           ;COMMAND FRAME ACK TIME OUT (HI BYTE)
1676 ;
1677 ;
1678 ;JTADRH  =      JTIMER/256  ;HI BYTE OF JUMP TIMER ROUTINE ADDR
1679 ;JTADRL  =      (-256)*JTADRH+JTIMER   ;"MOVED TO LINE 1428"
1680 ;
1681         .PAGE
1682 ;       SIO
1683 ;
1684 ;
1685         *=SIOV
1686         JMP     SIO         ;SIO ENTRY POINT
1687 ;
1688         *=SIOINV
1689         JMP     SIOINT      ;SIO INITIALIZATION ENTRY POINT
1690 ;
1691         *=SENDEV
1692         JMP     SENDEN      ;SEND ENABLE ENTRY POINT
1693 ;
1694         *=VCTABL-INTABS+VSERIN
1695 ;
1696         .WORD     ISRSIR    ;VSERIN
1697         .WORD     ISRODN    ;VSEROR
1698         .WORD     ISRTD     ;VSEROC
1699 ;
1700 ;
1701 ;
1702         *=SIOORG
1703 ;
1704 ; SIO INITIALIZATION SUBROUTINE
1705 ;
1706 SIOINT: LDA     #MOTRST
1707         STA     PACTL       ;TURN OFF MOTOR
```

```
1708 ;
1709         LDA     #NCOMHI
1710         STA     PBCTL       ;RAISE NOT COMMAND LINE
1711 ;
1712 ;
1713         LDA     #3
1714         STA     SSKCTL      ;GET POKEY OUT OF INITIALIZE MODE
1715         STA     SOUNDR      ;INIT POKE ADDRESS FOR QUIET I/O
1716         STA     SKCTL
1717 ;
1718 ;
1719         RTS                 ;RETURN
1720 ;
1721 ;
1722 ;
1723 ;
1724 ;
1725 ;
1726 SIO:    TSX
1727         STX     STACKP      ;SAVE STACK POINTER
1728         LDA     #1
1729         STA     CRITIC
1730 ;
1731         LDA     DDEVIC
1732         CMP     #CASET
1733         BNE     NOTCST      ;BRANCH IF NOT CASSETTE
1734         JMP     CASENT      ;OTHERWISE JUMP TO CASSETTE ENTER
1735 ;
1736 ; ALL DEVICES EXCEPT CASSETTE ARE INTELLIGENT
1737 ;
1738 NOTCST: LDA     #0
1739         STA     CASFLG      ; INIT CASSETTE FLAG TO NO CASSETTE
1740 ;
1741         LDA     #DRETRI     ;SET NUMBER OF DEVICE RETRIES
1742         STA     DRETRY
1743 COMMND: LDA     #CRETRI     ;SET NUMBER OF COMMAND FRAMERETRIES
1744         STA     CRETRY
1745 ;
1746 ; SEND A COMMAND FRAME
1747 ;
1748 COMFRM: LDA     #B192LO     ;SET BAUD RATE TO 19200
1749         STA     AUDF3
1750         LDA     #B192HI
1751         STA     AUDF4
1752 ;
1753         CLC                 ;SET UP COMMAND BUFFER
1754         LDA     DDEVIC
1755         ADC     DUNIT
1756         ADC     #$FF        ;SUBTRACT 1
1757         STA     CDEVIC      ;SET BUS ID NUMBER
1758 ;
1759         LDA     DCOMND
1760         STA     CCOMND      ;SET BUS COMMAND
1761 ;
1762         LDA     DAUX1       ;STORE COMMAND FRAME AUX BYTES 1 AND 2
1763         STA     CAUX1
1764         LDA     DAUX2
1765         STA     CAUX2       ;DONE SETTING UP COMMAND BUFFER
1766 ;
```

```
1767         CLC                    ;SET BUFFER POINTER TO COMMAND FRAME BUFFER
1768         LDA     #CBUFLO
1769         STA     BUFRLO         ;AND BUFFER END ADDRESS
1770         ADC     #4
1771         STA     BFENLO
1772         LDA     #CBUFHI
1773         STA     BUFRHI
1774         STA     BFENHI         ;DONE SETTING UP BUFFER POINTER
1775 ;
1776         LDA     #NCOMLO        ;LOWER NOT COMMAND LINE
1777         STA     PBCTL
1778 ;
1779         JSR     SENDIN         ;SEND THE COMMAND FRAME TO A SMART DEVICE
1780 ;
1781         LDA     ERRFLG
1782         BNE     BADCOM         ;BRANCH IF AN ERROR RECEIVED
1783 ;
1784         TYA
1785         BNE     ACKREC         ;BRANCH IF ACK RECEIVED
1786 ;
1787 ;
1788 BADCOM: DEC     CRETRY         ;A NACK OR TIME OUT OCCURED
1789         BPL     COMFRM         ;SO BRANCH IF ANY RETRIES LEFT
1790 ;
1791         JMP     DERR1          ;OTHERWISE, JUMP TO RETURN SECTION
1792 ;
1793 ;
1794 ACKREC: LDA     DSTATS         ;ACK WAS RECEIVED
1795         BPL     WATCOM         ;BRANCH TO WAIT FOR COMPLETE
1796 ; IF THERE IS NO DATA TO BE SENT
1797 ;
1798 ;
1799 ;
1800 ; SEND A DATA FRAME TO PERIPHERAL
1801 ;
1802         LDA     #CRETRI        ;SET NUMBER OF RETRIES
1803         STA     CRETRY
1804 ;
1805         JSR     LDPNTR         ;LOAD BUFFER POINTER WITH DCB INFORMATION
1806 ;
1807         JSR     SENDIN         ;GO SEND THE DATA FRAME TO A SMART DEVICE
1808 ;
1809         BEQ     BADCOM         ;BRANCH IF BAD
1810 ;
1811 ;
1812 ;
1813 ; WAIT FOR COMPLETE SIGNAL FROM PERIPHERAL
1814 ;
1815 WATCOM: JSR     STTMOT         ;SET DDEVICE TIME OUT VALUES IN Y,X
1816 ;
1817         LDA     #$00
1818         STA     ERRFLG         ;CLEAR ERROR FLAG
1819 ;
1820         JSR     WAITER         ;SET UP TIMER AND WAIT
1821         BEQ     DERR           ;BRANCH IF TIME OUT
1822 ;
1823 ;
1824 ; DEVICE DID NOT TIME OUT
1825 ;
```

```
1826         BIT     DSTATS
1827         BVS     MODATA      ;BRANCH IF MORE DATA FOLLOWS
1828 ;
1829         LDA     ERRFLG
1830         BNE     DERR1       ;BRANCH IF AN ERROR OCCURRED
1831         BEQ     RETURN      ;OTHERWISE RETURN
1832 ;
1833 ;
1834 ;
1835 ;
1836 ; RECEIVE A DATA FRAME FROM PERIPHERAL
1837 ;
1838 MODATA: JSR     LDPNTR      ;LOAD BUFFER POINTER WITH DCB INFORMATION
1839 ;
1840         JSR     RECEIV      ;GO RECEIVE A DATA FRAME
1841 ;
1842 DERR:   LDA     ERRFLG
1843         BEQ     NOTERR      ;BRANCH IF NO ERROR PRECEEDED DATA
1844 ;
1845         LDA     TSTAT       ;GET TEMP STATUS
1846         STA     STATUS      ;STORE IN REAL STATUS
1847 ;
1848 ;
1849 NOTERR: LDA     STATUS
1850         CMP     #SUCCES
1851         BEQ     RETURN      ;BRANCH IF COMPLETELY SUCCESSFUL
1852 ;
1853 DERR1:  DEC     DRETRY
1854         BMI     RETURN      ;BRANCH IF OUT OF DEVICE RETRIES
1855 ;
1856         JMP     COMMND      ;OTHERWISE ONE MORE TIME
1857 ;
1858 ;
1859 ;
1860 ;
1861 RETURN: JSR     SENDDS      ;DISABLE POKEY INTERRUPTS
1862         LDA     #0
1863         STA     CRITIC
1864         LDY     STATUS      ;RETURN STATUS IN Y
1865         STY     DSTATS      ;AND THE DCB STATUS WORD
1866         RTS     RETURN
1867 ;
1868 ;
1869 ;
1870 ;
1871 ; WAIT SUBROUTINE
1872 ;
1873 ; WAITS FOR COMPLETE OR ACK
1874 ; RETURNS Y=$FF IF SUCCESSFUL, Y=$00 IF NOT
1875 ;
1876 WAIT:   LDA     #$00
1877         STA     ERRFLG      ;CLEAR ERROR FLAG
1878 ;
1879         CLC                 ;LOAD BUFFER POINTER WITH ADDRESS
1880         LDA     #TEMPLO     ;OF TEMPORARY RAM CELL
1881         STA     BUFRLO
1882         ADC     #1
1883         STA     BFENLO      ;ALSO SET BUFFER END +1 ADDRESS
1884         LDA     #TEMPHI
```

```
1885         STA     BUFRHI
1886         STA     BFENHI      ;DONE LOADING POINTER
1887 ;
1888         LDA     #$FF
1889         STA     NOCKSM      ;SET NO CHECKSUM FOLLOWS DATA FLAG
1890 ;
1891         JSR     RECEIV      ;GO RECEIVE A BYTE
1892 ;
1893         LDY     #$FF        ;ASSUME SUCCESS
1894         LDA     STATUS
1895         CMP     #SUCCES
1896         BNE     NWOK        ;BRANCH IF IT DID NOT WORK OK
1897 ;
1898 ;
1899 ;
1900 ;
1901 WOK:    LDA     TEMP        ;MAKE SURE THE BYTE SUCCESSFULLY RECEIVED
1902         CMP     #ACK        ;WAS ACTUALLY AN ACK OR COMPLETE
1903         BEQ     GOOD
1904         CMP     #COMPLT
1905         BEQ     GOOD
1906 ;
1907         CMP     #ERROR
1908         BNE     NOTDER      ;BRANCH IF DEVICE DID NOT SEND BACK
1909 ; A DEVICE ERROR CODE
1910         LDA     #DERROR
1911         STA     STATUS      ;SET DEVICE ERROR STATUS
1912         BNE     NWOK
1913 ;
1914 NOTDER: LDA     #DNACK      ;OTHERWISE SET HACK STATUS
1915         STA     STATUS
1916 ;
1917 NWOK:   LDA     STATUS
1918         CMP     #TIMOUT
1919         BEQ     BAD         ;BRANCH IF TIME OUT
1920 ;
1921         LDA     #$FF
1922         STA     ERRFLG      ;SET SOME ERROR FLAG
1923         BNE     GOOD        ;RETURN WITH OUT SETTING Y = 0
1924 ;
1925 BAD:    LDY     #0
1926 ;
1927 GOOD:   LDA     STATUS
1928         STA     TSTAT
1929         RTS                 ;RETURN
1930 ;
1931 ;
1932 ;
1933 ;
1934 ;
1935 ; SEND SUBROUTINE
1936 ;
1937 ; SENDS A BUFFER OF BYTES OUT OVER THE SERIAL BUS
1938 ;
1939 ;
1940 SEND:   LDA     #SUCCES     ;ASSUME SUCCESS
1941         STA     STATUS
1942 ;
1943         JSR     SENDEN      ;ENABLE SENDING
```

```
1944 ;
1945         LDY     #0
1946         STY     CHKSUM      ;CLEAR CHECK SUM
1947         STY     CHKSNT      ;CHECKSUM SENT FLAG
1948         STY     XMTDON      ;TRANSMISSION DONE FLAG
1949 ;
1950 ;
1951         LDA     (BUFRLO),Y  ;PUT FIRST BYTE FROM BUFFER
1952         STA     SEROUT      ;INTO THE SERIAL OUTPUT REGISTER
1953 ;
1954 ;
1955         STA     CHKSUM      ;PUT IT IN CHECKSUM
1956 ;
1957 NOTDON: LDA     BRKKEY
1958         BNE     NTBRKO
1959         JMP     BROKE       ;JUMP IF BREAK KEY PRESSED
1960 ;
1961 NTBRKO: LDA     XMTDON      ;LOOP UNTIL TRANSMISSION IS DONE
1962         BEQ     NOTDON
1963 ;
1964         JSR     SENDDS      ;DISABLE SENDING
1965 ;
1966         RTS     RETURN
1967 ;
1968 ;
1969 ;
1970 ;
1971 ;
1972 ;
1973 ; OUTPUT DATA NEEDED INTERRUPT SERVICE ROUTINE
1974 ;
1975 ISRODN: TYA
1976         PHA                 ;SAVE Y REG ON STACK
1977 ;
1978         INC     BUFRLO      ;INCREMENT DUFFER POINTER
1979         BNE     NOWRPO
1980         INC     BUFRHI
1981 ;
1982 NOWRPO: LDA     BUFRLO      ;CHECK IF PAST END OF BUFFER
1983         CMP     BFENLO
1984         LDA     BUFRHI      ;HIGH PART
1985         SBC     BFENHI
1986         BCC     NOTEND      ;BRANCH IF NOT PAST END OF BUFFER
1987 ;
1988         LDA     CHKSNT
1989         BNE     RELONE      ;BRANCH IF CHECKSUM ALREADY SENT
1990 ;
1991         LDA     CHKSUM
1992         STA     SEROUT      ;SEND CHECK SUM
1993         LDA     #$FF
1994         STA     CHKSNT      ;SET CHECKSUM SENT FLAG
1995         BNE     CHKDON
1996 ;
1997 RELONE: LDA     POKMSK      ;ENABLE TRANSMIT DONE INTERRUPT
1998         ORA     #$08
1999         STA     POKMSK
2000         STA     IRQEN
2001 ;
2002 CHKDON: PLA
```

```
2003         TAY                  ;RESTORE Y REG
2004         PLA                  ;RETURN FROM INTERRUPT
2005         RTI
2006 ;
2007 ;
2008 NOTEND: LDY     #0
2009         LDA     (BUFRLO),Y  ;PUT NEXT BYTE FROM BUFFER
2010         STA     SEROUT      ;INTO THE SERIAL OUTPUT REGISTER
2011 ;
2012         CLC                  ;ADD IT TO CHECKSUM
2013         ADC     CHKSUM
2014         ADC     #0
2015         STA     CHKSUM
2016 ;
2017         JMP     CHKDON      ;GO RETURN
2018 ;
2019 ;
2020 ;
2021 ;
2022 ;
2023 ;
2024 ; TRANSMIT DONE INTERRUPT SERVICE ROUTINE
2025 ;
2026 ISRTD:  LDA     CHKSNT
2027         BEQ     FOOEY       ;BRANCH IF CHECKSUM NOT YET SENT
2028 ;
2029         STA     XMTDON      ;OTHERWISE SET TRANSMISSION DONE FLAG
2030 ;
2031         LDA     POKMSK      ;DISABLE TRANSMIT DONE INTERRUPT
2032         AND     #$F7
2033         STA     POKMSK
2034         STA     IRQEN
2035 ;
2036 FOOEY:  PLA                  ;RETURN FROM INTERRUPT
2037         RTI
2038 ;
2039 ;
2040 ;
2041 ;
2042 ;
2043 ;
2044 ;
2045 ;
2046 ; RECEIVE SUBROUTINE
2047 ;
2048 RECEIV: LDA     #0
2049 ;
2050         LDY     CASFLG
2051         BNE     NOCLR       ;BRANCH IF CASSETTE
2052 ;
2053         STA     CHKSUM      ;CLEAR CHKSUM
2054 NOCLR:  STA     BUFRFL      ;BUFFER FULL FLAG
2055         STA     RECVDN      ;RECEIVE DONE FLAG
2056 ;
2057 ;
2058 ;
2059         LDA     #SUCCES
2060         STA     STATUS      ;SET GOOD STATUS FOR DEFAULT CASE.
2061         JSR     RECVEN      ;DO RECEIVE ENABLE
```

```
2062         LDA     #NCOMHI     ;COMMAND FRAME HI COMMAND
2063         STA     PBCTL       ;STORE IN PIA
2064 CHKTIM: LDA     BRKKEY
2065         BNE     NTBRK1
2066         JMP     BROKE       ;JUMP IF BREAK KEY PRESSED
2067 ;
2068 NTBRK1: LDA     TIMFLG      ;NO,
2069         BEQ     TOUT        ;IF TIMEOUT, GO SET ERROR STATUS
2070         LDA     RECVDN
2071         BEQ     CHKTIM      ;DONE ?
2072 GOBACK: RTS
2073 TOUT:   LDA     #TIMOUT     ;YES,
2074         STA     STATUS      ;SET TIMEOUT STATUS
2075 ;
2076 ;
2077 ;
2078 ;
2079 ;
2080 ;
2081 RRETRN: RTS                 ;RETURN
2082 ;
2083 ;
2084 ;
2085 ;
2086 ;
2087 ;
2088 ;
2089 ; SERIAL INPUT READY INTERRUPT SERVICE ROUTINE
2090 ;
2091 ISRSIR: TYA
2092         PHA                 ;SAVE Y REG ON STACK
2093 ;
2094 ;
2095 ;
2096         LDA     SKSTAT
2097         STA     SKRES       ;RESET STATUS REGISTER
2098 ; ********   THIS MAY NOT BE THE PLACE TO DO IT   *********
2099 ;
2100         BMI     NTFRAM      ;BRANCH IF NO FRAMING ERROR
2101 ;
2102         LDY     #FRMERR
2103         STY     STATUS      ;SET FRAME ERRORR STATUS
2104 ;
2105 NTFRAM: AND     #$20
2106         BNE     NTOVRN      ;BRANCH IF NO OVERRUN ERROR
2107 ;
2108         LDY     #OVRRUN
2109         STY     STATUS      ;SET OVERRUN ERROR STATUS
2110 ;
2111 NTOVRN: LDA     BUFRFL
2112         BEQ     NOTYET      ;BRANCH IF BUFFER WAS NOTYET FILLED
2113 ;
2114         LDA     SERIN       ;THIS INPUT BYTE 15 THE CHECKSUM
2115         CMP     CHKSUM
2116         BEQ     SRETRN      ;BRANCH IF CHECKSUMS MATCH
2117 ;
2118         LDY     #CHKERR
2119         STY     STATUS      ;SET CHECKSUM ERROR STATUS
2120 ;
```

```
2121 SRETRN: LDA     #$FF         ;SET RECEIVE DONE FLAG
2122         STA     RECVDN
2123 ;
2124 SUSUAL: PLA
2125         TAY                  ;RESTORE Y REG
2126         PLA                  ;RETURN FROM INTERRUPT
2127         RTI
2128 ;
2129 ;
2130 ;
2131 NOTYET: LDA     SERIN
2132         LDY     #0
2133         STA     (BUFRLO),Y   ;STORE INPUT REGISTER INTO BUFFER
2134 ;
2135         CLC                  ;ADD IT TO CHECKSUM
2136         ADC     CHKSUM
2137         ADC     #0
2138         STA     CHKSUM
2139 ;
2140         INC     BUFRLO       ;INCREMENT BUFFER POINTER
2141         BNE     NTWRP1
2142         INC     BUFRHI
2143 ;
2144 NTWRP1: LDA     BUFRLO
2145         CMP     BFENLO
2146         LDA     BUFRHI
2147         SBC     BFENHI
2148         DCC     SUSUAL       ;BRANCH IF NEW BUFFER ADDRESS IS IN BUFFER L
2149 ;
2150         LDA     NOCKSM
2151         BEQ     GOON         ;BRANCH IF A CHECKSUM WILL FOLLOW DATA
2152 ;
2153         LDA     #0
2154         STA     NOCKSM       ;CLEAR NO CHECKSUM FLAG
2155 ;
2156         BEQ     SRETRN       ;GO RETURN AND SET RECEIVE DONE FLAG
2157 ;
2158 ;
2159 GOON:   LDA     #$FF
2160         STA     BUFRFL       ;SET BUFFER FULL FLAG
2161 ;
2162         BNE     SUSUAL       ;GO RETURN
2163 ;
2164 ;
2165 ;
2166 ;
2167 ;
2168 ;
2169 ;
2170 ;
2171 ; LOAD BUFFER POINTER SUBROUTINE
2172 ;
2173 ; LOAD BUFFER POINTER WITH DCB BUFFER INFORMATION
2174 ;
2175 LDPNTR: CLC
2176         LDA     DBUFLO
2177         STA     BUFRLO
2178         ADC     DBYTLO
2179         STA     BFENLO       ;ALSO SET SUFFER END + 1 ADDRESS
```

```
2180 ;
2181         LDA     DBUFHI
2182         STA     BUFRHI
2183         ADC     DBYTHI
2184         STA     BFENHI
2185 ;
2186         RTS                 ;RETURN
2187 ;
2188 ;
2189 ;
2190 ;
2191 ;
2192 ;
2193 ;
2194 ;
2195 ; CASSETTE HANDLING CODE
2196 ;
2197 CASENT: LDA     DSTATS
2198         BPL     CASRED      ;BRANCH IF INPUT FROM CASSETTE
2199 ;
2200 ; WRITE A RECORD
2201 ;
2202         LDA     #B600LO     ;SET BAUD RATE TO 600
2203         STA     AUDF3
2204         LDA     #B600HI
2205         STA     AUDF4
2206 ;
2207         JSR     SENDEN      ;TURN ON POKEY MARK TONE
2208 ;
2209         LDY     #WSIRG      ;LOAD SHORT WRITE INTER RECORD GAP TIME
2210         LDA     DAUX2
2211         BMI     SRTIR0      ;BRANCH IF SHORT GAP IS DESIRED
2212 ;
2213         LDY     #WIRGLO     ;SET WRITE IRQ TIME
2214 SRTIR0: LDX     #WIRGHI
2215         JSR     SETVBX
2216 ;
2217         LDA     #MOTRGO
2218         STA     PACTL       ;TURN ON MOTOR
2219 ;
2220 TIMIT:  LDA     TIMFLG      ;LOOP UNTIL DONE
2221         BNE     TIMIT
2222 ;
2223         JSR     LDPNTR      ;LOAD BUFFER POINTER WITH DCB INFORMATION
2224 ;
2225         JSR     SEND        ;SEND A BUFFER
2226 ;
2227         JMP     CRETRN      ;GO, RETURN
2228 ;
2229 ;
2230 ;
2231 ; RECEIVE A RECORD
2232 ;
2233 CASRED: LDA     #$FF
2234         STA     CASFLG      ;SET SET CASSETTE FLAG
2235 ;
2236         LDY     #RSIRG      ;LOAD SHORT READ INTER RECORD GAP TIME
2237         LDA     DAUX2
2238         BMI     SRTIR1      ;BRANCH IF SHORT GAP IS DESIRED
```

```
2239 ;
2240         LDY     #RIRGLO     ;SET TIME OUT FOR READ IRQ
2241 SRTIR1: LDX     #RIRGHI
2242         JSR     SETVBX
2243 ;
2244         LDA     #MOTRGO
2245         STA     PACTL       ;TURN ON MOTOR
2246 ;
2247 TIMIT1: LDA     TIMFLG      ;LOOP UNTIL DONE
2248         BNE     TIMIT1
2249 ;
2250         JSR     LDPNTR      ;LOAD BUFFER POINTER WITH DOS INFORMATION
2251 ;
2252         JSR     STTMOT      ;SET DEVICE TIME OUT IN Y,X
2253         JSR     SETVBX
2254 ;
2255         JSR     BEGIN       ;SET INITIAL BAUD RATE
2256 ;
2257         JSR     RECEIV      ;GO RECEIVE A BLOCK
2258 ;
2259 CRETRN: LDA     DAUX2
2260         BMI     SRTIR2      ;BRANCH IF DOING SHORT INTER RECORD GAPS
2261 ; DON'T TURN OFF CASSETTE MOTOR
2262         LDA     #MOTRST
2263         STA     PACTL       ;TURN OFF MOTOR
2264 ;
2265 SRTIR2: JMP     RETURN      ;GO RETURN
2266 ;
2267 ;
2268 ;
2269 ;
2270 ;
2271 JTIMER: LDA     #$00
2272 JTADRH  =       JTIMER/256  ;HI BYTE OF JUMP TIMER ROUTINE ADDR
2273 JTADRL  =       (-256)*JTADRH+JTIMER
2274         STA     TIMFLG      ;SET TIME OUT FLAG
2275         RTS
2276 ;
2277 ;
2278 ;
2279 ;
2280 ;
2281 ;
2282 ; SEND ENABLE SUBROUTINE
2283 ;
2284 SENDEN: LDA     #$07        ;MASK OFF PREVIOUS SERIAL BUS CONTROL BITS
2285         AND     SSKCTL
2286         ORA     #$20        ;SET TRANSMIT MODE
2287 ;
2288         LDY     DDEVIC
2289         CPY     #CASET
2290         BNE     NOTCAS      ;BRANCH IF NOT CASSETTE
2291 ;
2292         ORA     #$08        ;SET THE FSK OUTPUT BIT
2293 ;
2294         LDY     #LOTONE     ;SET FSK TONE FREQUENCIES
2295         STY     AUDF2
2296         LDY     #HITONE
2297         STY     AUDF1
```

```
2298 ;
2299 NOTCAS: STA     SSKCTL      ;STORE NEW VALUE TO SYSTEM MASK
2300         STA     SKCTL       ;STORE TO ACTUAL REGISTER
2301 ;
2302         LDA     #$C7        ;MASK OFF PREVIOUS SERIAL BUS INTERRUPT BITS
2303         AND     POKMSK
2304         ORA     #$10        ;ENABLE OUTPUT DATA NEEDED INTERRUPT
2305 ;
2306 ;
2307         JMP     CONTIN      ;GO CONTINUE IN RECEIVE ENABLE SUBROUTINE
2308 ;
2309 ;
2310 ;
2311 ;
2312 ;
2313 ;
2314 ;
2315 ;
2316 ;
2317 ;
2318 ; RECEIVE ENABLE SUBROUTINE
2319 ;
2320 RECVEN: LDA     #$07        ;MASK OFF PREVIOUS SERIAL BUS CONTROL BITS
2321         AND     SSKCTL
2322         ORA     #$10        ;SET RECEIVE MODE ASYNCH.
2323         STA     SSKCTL      ;STORE NEW VALUE TO SYSTEM MASK
2324         STA     SKCTL       ;STORE TO ACTUAL REGISTER
2325 ;
2326         STA     SKRES       ;RESET SERIAL PORT/KEYBOARD STATUS REGISTER
2327 ;
2328         LDA     #$C7        ;MASK OFF PREVIOUS SERIAL BUS INTERRUPTBITS
2329         AND     POKMSK
2330         ORA     #$20        ;ENABLE RECEIVE INTERRUPT
2331 CONTIN: STA     POKMSK      ;STORE NEW VALUE TO SYSTEM MASK
2332         STA     IRQEN       ;STORE TO ACTUAL. REGISTER
2333 ;
2334 ;
2335         LDA     #$28        ;CLOCK CH.3 WITH 1.79 MHZ
2336         STA     AUDCTL      ;CLOCK CH.4 WITH CH. 3
2337 ;
2338         LDX     #6          ;SET PURE TONES, NO VOLUME
2339         LDA     #$A8
2340         LDY     SOUNDR      ;TEST QUIET I/O FLAG
2341         BNE     NOISE1      ;NE IS NORMAL (NOISY)
2342         LDA     #$A0
2343 NOISE1: STA     AUDC1,X
2344         DEX
2345         DEX
2346         BPL     NOISE1
2347 ;
2348         LDA     #$A0
2349         STA     AUDC3       ;TURN OFF SOUND ON CHANNEL 3
2350         LDY     DDEVIC
2351         CPY     #CASET
2352         BEQ     CAS31       ;BRANCH IF CASSETTE IS DESIRED
2353         STA     AUDC1       ;OTHERWISE TURN OFF CHANNELS 1 AND 2
2354         STA     AUDC2
2355 ;
2356 ;
```

```
2357 CAS31:   RTS               ;RETURN
2358 ;
2359 ;
2360 ;
2361 ;
2362 ;
2363 ;
2364 ;
2365 ;
2366 ;
2367 ;
2368 ; DISABLE SEND AND DISABLE RECEIVE SUBROUTINES
2369 ;
2370 SENDDS: NOP
2371 RECVDS: LDA     #$C7        ;MASK OFF SERIAL BUS INTERRUPTS
2372         AND     POKMSK
2373         STA     POKMSK      ;STORE NEW VALUE TO SYSTEM MASK
2374         STA     IRQEN       ;STORE TO ACTUAL REGISTER
2375 ;
2376         LDX     #6
2377         LDA     #0
2378 ZERIT:  STA     AUDC11X
2379         DEX
2380         DEX
2381         BPL     ZERIT       ;TURN OFF AUDIO VOLUME
2382 ;
2383         RTS                 ;RETURN
2384 ;
2385 ;
2386 ;
2387 ;
2388 ;
2389 ;
2390 ;
2391 ;
2392 ;
2393 ;
2394 ; SET DDEVICE TIME OUT VALUES IN Y,X SUBROUTINE
2395 ;
2396 STTMOT: LDA     DTIMLO      ;GET DEVICE TIME OUT IN 1 SECOND INCR
2397         ROR     A           ;PUT 6 HI BITS IN X, LO 2 BITS IN Y
2398         ROR     A
2399         TAY                 ;TEMP SAVE
2400         AND     #$3F        ;MASK OFF 2 HI BITS
2401         TAX                 ;THIS IS HZ BYTE OF TIME OUT
2402 ;
2403         TYA                 ;RESTORE
2404         ROR     A
2405         AND     #$C0        ;MASK OFF ALL BUT 2 HI BITS
2406         TAY                 ;THIS IS LO BYTE OF TIME OUT
2407 ;
2408         RTS
2409 ;
2410 ;
2411 ;
2412 ;
2413 ;
2414 ;
2415 ;
```

```
2416 ;
2417 ;
2418 ;
2419 INTTBL: .WORD   ISRSIR      ;SERIAL INPUT READY
2420         .WORD   ISRODN      ;OUTPUT DATA NEEDED
2421         .WORD   ISRTD       ;TRANSMISSION DONE
2422 ;
2423 SIRHI  =       ISRSIR/256   ;SERIAL INPUT READY ISR ADDRESS
2424 SIRLO  =       (-256)*SIRHI+ISRSIR
2425 ODNHI  =       ISRODN/256   ;OUTPUT DATA NEEDED ISR ADDRESS
2426 ODNLO  =       (-256)*ODNHI+ISRODN
2427 TDHI   =       ISRTD/256    ;TRANSMISSION DONE ISR ADDRESS
2428 TDLO   =       (-256)*TDHI+ISRTD
2429 ;
2430 ;
2431 ;
2432 ;
2433 ; SEND A DATA FRAME TO AN INTELLIGENT PERIPHERAL SUBROUTINE
2434 ;
2435 ;
2436 SENDIN: LDX     #$01
2437 DELAY0: LDY     #$FF
2438 DELAY1: DEY
2439         BNE     DELAY1
2440         DEX
2441         BNE     DELAY0
2442 ;
2443         JSR     SEND        ;GO SEND THE DATA FRAME
2444 ;
2445         LDY     #CTIMLO     ;SET ACK TIME OUT
2446         LDX     #CTIMHI
2447 WAITER: JSR     SETVBX
2448
2449         JSR     WAIT        ;WAIT FOR ACK
2450 ;
2451         TYA                 ;IF Y=0, A TIME OUT OR NACK OCCURED
2452 ;
2453         RTS                 ;RETURN
2454 ;
2455 ;
2456 ;
2457 ;
2458 ;
2459 ;
2460 ;
2461 ;
2462 ;
2463 ;
2464 ;
2465 ; COMPUTE VALUE FOR POKEY FREQ REGS FOR THE BAUD RATE AS
2466 ; MEASURED BY AN INTERVAL OF THE 'VCOUNT' TIMER.
2467 ;
2468 COMPUT: STA     TIMER2
2469         STY     TIMER2+1    ;SAVE FINAL TIMER VALUE
2470         JSR     ADJUST      ;ADJUST VCOUNT VALUE
2471         STA     TIMER2      ;SAVE ADJUSTED VALUE
2472         LDA     TIMER1
2473         JSR     ADJUST      ;ADJUST
2474         STA     TIMER1      ;SAVE ADJUSTED TIMER1 VALUE
```

```
2475         LDA     TIMER2
2476         SEC
2477         SBC     TIMER1
2478         STA     TEMP1       ;FIND VCOUNT DIFFERENCE
2479         LDA     TIMER2+1
2480         SEC
2481         SBC     TIMER1+1
2482         TAY                 ;FIND VBLANK COUNT DIFFERENCE
2483         .IF     PALFLG
2484         LDA     #-$9C
2485 HITIMR: CLC
2486         ADC     #$9C
2487         .ENDIF
2488         .IF     PALFLG-1
2489         LDA     #-$83
2490 HITIMR: CLC
2491         ADC     #$83        ;ACCUMULATE MULTIPLICATION
2492         .ENDIF
2493         DEY
2494         BPL     HITIMR      ;DONE?
2495         CLC
2496         ADC     TEMP1       ;TOTAL VCOUNT DIFFERENCE
2497 FINDX:  TAY                 ;SAVE ACCUM
2498         LSR     A
2499         LSR     A
2500         LSR     A
2501         ASL     A
2502         SEC
2503         SBC     #22         ;ADJUST TABLE INDEX
2504         TAX                 ;DIVIDE INTERVAL BY 4 TO CET TABLE INDEX
2505         TYA                 ;RESTORE ACCUM
2506         AND     #7
2507         TAY                 ;PULL OFF 3 LO BITS OF INTERVAL
2508         LDA     #-11
2509 DOINTP: CLC
2510         ADC     #11         ;ACCUMULATE INTERPOLATION CONSTANT
2511         DEY
2512         BPL     DOINTP      ;INTERPOLATION CONSTANT COMPUTATION DONE?
2513 ;
2514 ENINTP: LDY     #0
2515         STY     ADDCOR      ;CLEAR ADDITION CORRECTION FLAG
2516         SEC
2517         SBC     #7          ;ADJUST INTERPOLATION CONSTANT
2518         BPL     PLUS
2519         DEC     ADDCOR
2520 PLUS:   CLC
2521         ADC     POKTAB,X    ;ADD CONSTANT TO LO BYTE TABLE VALUE
2522         TAY                 ;LO BYTE POKEY FREQ VALUE
2523         LDA     ADDCOR
2524         ADC     POKTAB+1,X  ;ADD CARRY TO HI BYTE TABLEVALUE
2525 ; HI BYTE POKEY FREQ VALUE
2526         RTS
2527 ;
2528 ;
2529 ;
2530 ;       ROUTINE TO ADJUST VCOUNT VALUE
2531 ;
2532 ADJUST: CMP     #$7C
2533         BMI     ADJ1        ;LAROER THAN '7C' ?
```

```
2534         SEC                    ;YES
2535         SBC     #$7C
2536         RTS
2537 ADJ1:   CLC
2538         .IF     PALFLG
2539         ADC     #$20
2540         .ENDIF
2541         .IF     PALFLG-1
2542         ADC     #$7
2543         .ENDIF
2544         RTS
2545 ;
2546 ;
2547 ;
2548 ;
2549 ;
2550 ;
2551 ;
2552 ;       INITIAL BAUD RATE MEASUREMENT -- USED TO SET THE
2553 ;               BAUD RATE AT THE START OF A RECORD.
2554 ;
2555 ;               IT IS ASSUMED THAT THE FIRST TWO BYTES OF EVERY
2556 ; RECORD ARE 'AA' HEX.
2557 ;
2558 BEGIN:  LDA     BRKKEY
2559         BNE     NTBRK2
2560         JMP     BROKE          ;JUMP IF BREAK KEY PRESSED
2561 ;
2562 NTBRK2: SEI
2563 ;
2564         LDA     TIMFLG
2565         BNE     OKTIM1         ;BRANCH IF NOT TIMEDOUT
2566         BEQ     TOUT1          ;BRANCH IF TIME OUT
2567 ;
2568 OKTIM1: LDA     SKSTAT
2569         AND     #$10           ;READ SERIAL PORT
2570         BNE     BEGIN          ;START BIT?
2571         STA     SAVIO          ;SAVE SER. DATA IN
2572         LDX     VCOUNT         ;READ VERTICAL LINECOUNTER
2573         LDY     RTCLOK+2       ;READ LO BYTE OF VBLANK CLOCK
2574         STX     TIMER1
2575         STY     TIMER1+1       ;SAVE INITIAL TIMER VALUE
2576 ;
2577         LDX     #1             ;SET MODE FLAG
2578         STX     TEMP3
2579         LDY     #10            ;SET BIT COUNTER FOR 10 BITS
2580 COUNT:  LDA     BRKKEY
2581         BEQ     BROKE          ;BRANCH IF BREAK KEY PRESSED
2582 ;
2583         LDA     TIMFLG
2584         BNE     OKTIMR         ;BRANCH IF NOT TIMED OUT
2585 TOUT1:  CLI
2586         JMP     TOUT           ;BRANCH IF TIME OUT
2587 ;
2588 OKTIMR: LDA     SKSTAT
2589         AND     #$10           ;READ SERIAL PORT
2590         CMP     SAVIO          ;DATA IN CHANGED YET?
2591         BEQ     COUNT
2592         STA     SAVIO          ;YES,SAVE SER. DATA IN
```

```
2593         DEY                 ;DECR. BIT COUNTER
2594         BNE     COUNT       ;DONE?
2595 ;
2596         DEC     TEMP3       ;YES,
2597         BMI     GOREAD      ;DONE WITH BOTH MODES?
2598         LDA     VCOUNT
2599         LDY     RTCLOK+2    ;READ TIMER LO & HI BYTES
2600         JSR     COMPUT      ;NO, COMPUTE SAUD RATE
2601         STY     CBAUDL
2602         STA     CBAUDH      ;SET BAUD RATE INTO RAM CELLS
2603         LDY     #9          ;SET BIT COUNTER FOR 9 BITS
2604         BNE     COUNT
2605 ;
2606 GOREAD: LDA     CBAUDL
2607         STA     AUDF3
2608         LDA     CBAUDH
2609         STA     AUDF4       ;SET POKEY FREQ REGS FOR BAUD RATE
2610         LDA     #0
2611         STA     SKSTAT
2612         LDA     SSKCTL
2613         STA     SKSTAT      ; INIT. POKEY SERIAL PORT
2614         LDA     #$55
2615         STA     (BUFRLO),Y  ;STORE '$55' AS FIRST RCV. BUFFER
2616         INY
2617         STA     (BUFRLO),Y
2618         LDA     #$AA
2619         STA     CHKSUM      ;STORE CHECKSUM FOR 2 BYTES OF '$AA'
2620         CLC
2621         LDA     BUFRLO
2622         ADC     #2
2623         STA     BUFRLO
2624         LDA     BUFRHI
2625         ADC     #0
2626         STA     BUFRHI      ;INCR. BUFFER POINTER BY 1
2627         CLI
2628         RTS
2629 ;
2630 ;
2631 ;
2632 BROKE:  JSR     SENDDS      ;BREAK KEY WAS PRESSED, SO PREPARE
2633         LDA     #MOTRST     ;TO RETURN
2634         STA     PACTL       ;TURN OFF MOTOR
2635         STA     PBCTL       ;RAISE NOT COMMAND LINE
2636 ;
2637         LDA     #BRKABT
2638         STA     STATUS      ;STORE BREAK ABORT STATUS CODE
2639 ;
2640         LDX     STACKP
2641         TXS                 ;RESTORE STACK POINTER
2642 ;
2643         DEC     BRKKEY      ;SET BREAK KEY FLAG TO NONZERO
2644         CLI                 ;ALLOW IRQ'S
2645 ;
2646         JMP     RETURN      ;GO RETURN
2647 ;
2648 ;
2649 ;
2650 ;
2651 ;
```

```
2652 SETVBX: LDA     #JTADRL     ;STORE TIME OUT ROUTINE ADDRESS
2653         STA     CDTMA1
2654         LDA     #JTADRH
2655         STA     CDTMA1+1
2656 ;
2657         LDA     #1          ;SET FOR TIMER 1
2658 ;
2659         SEI                 ;THE SETVBL ROUTINE NEEDS THIS TO CUT SHORT
2660         JSR     SETVBV      ;ANY VBLANKS THAT OCCUR
2661         LDA     #1          ;SET FOR TIMER 1
2662         STA     TIMFLG      ;SET FLAG TO NOT TIMED OUT
2663         CLI
2664         RTS
2665 ;
2666 ;
2667 ;
2668 ;
2669 ;
2670 ;
2671 ;
2672 ; 'VCOUNT' INTERVAL TIMER MEASUREMENT -- TO -- POKEY FREQ REG VALUE
2673 ;               CONVERSION TABLE
2674 ;
2675 ;
2676 ; THE VALUES STORED IN THE TABLE ARE 'AUDF+7'.
2677 ;
2678 ;       THE FOLLOWING FORMULAS WERE USED TO DETERMINE THE TABLE VALUES:
2679 ;
2680 ;               F OUT F IN/(2*(AUDF+M)) , WHERE F IN=1.78979 MHZ. & M=7
2681 ;
2682 ;        FROM THIS WAS DERIVED THE FORMULA USED TO COMPUTE THE
2683 ;        TABLE VALUES BASED ON A MEASUREMENT OF THE PERIOD BY
2684 ;        AN INTERVAL OF THE 'VCOUNT' TIMER.
2685 ;
2686 ;               AUDF+7=(11.365167)*T OUT, WHERE T OUT=# OF COUNTS
2687 ;               (127 USEC. RESOLUTION) OF 'VCOUNT' FOR 1
2688 ;               CHARACTER TIME (10 BIT TIMES).
2689 ;
2690 ;
2691 ;
2692 ;
2693 ;               AUDF+7          BAUD RATE        VCOUNT INTERVAL
2694 ;               ------          -----------      ----------------
2695 ;       .WORD   $27C            ;1407                    56
2696 ;       .WORD   $2D7            ;1231                    64
2697 ;       .WORD   $332            ;1094                    72
2698 ;       .WORD   $38D            ;985                     80
2699 POKTAB: .WORD   $3E8            ;895                     88
2700        .WORD   $443            ;820                     96
2701        .WORD   $49E            ;757                     104
2702        .WORD   $4F9            ;703                     112
2703        .WORD   $554            ;656                     120
2704        .WORD   $5AF            ;615                     128
2705        .WORD   $60A            ;579                     136
2706        .WORD   $665            ;547                     144
2707        .WORD   $6C0            ;518                     152
2708        .WORD   $71A            ;492                     160
2709        .WORD   $775            ;469                     168
2710        .WORD   $7D0            ;447                     176
```

```
2711 ;       .WORD   $828           ;428                     184
2712 ;       .WORD   $886           ;410                     192
2713 ;       .WORD   $8E1           ;394                     200
2714 ;       .WORD   $93C           ;379                     208
2715 ;       .WORD   $997           ;365                     216
2716 ;       .WORD   $9F2           ;352                     224
2717 ;       .WORD   $A4D           ;339                     232
2718 ;       .WORD   $AA8           ;328                     240
2719 ;       .WORD   $803           ;318                     248
2720 ;
2721 ;
2722 ;
2723 ;
2724 ;
2725 CRNTP3  =*
2726         *=$14
2727 SIOSPR: .BYTE   DSKORG-CRNTP3 ;^GSIOL IS TOO LONG
2728 ;
2729         .TITLE  'DISK  ***** DISKP.SRC ***** 3/9/79 *****  4:00:00 P.M.'
2730 ;
2731 ;
2732 ;
2733 ;
2734 ;
2735 ;
2736 STATVH  =       DVSTAT/256
2737 STATVL  =       (-256)*STATVH+DVSTAT ;STATUS POINTER
2738 ;
2739 ;
2740 ;
2741 ;
2742 ;       CONSTANT EQUATES
2743 ;
2744 DISKID  =       $31         ;SERIAL BUS DISK I.D.
2745 PUTSEC  =       $50         ;DISK PUT SECTOR DCB COMMAND
2746 ; READ  =       $52         ;DISK GET SECTOR DCB COMMAND
2747 ; WRITE =       $57         ;DISK PUT SECTOR WITH READ CHECK DCB COMMAND
2748 STATC   =       $53         ;DISK STATUS DCB COMMAND
2749 FOMAT   =       $21         ;DISK FORMAT DCB COMMAND !!!!! *****
2750 NODAT   =       0           ;SIO COMMAND FOR "NO DATA" OPERATION
2751 GETDAT  =       $40         ;SIO COMMAND FOR "DATA FROM DEVICE"
2752 PUTDAT  =       $80         ;SIO COMMAND FOR "DATA TO DEVICE"
2753 ;
2754 ;
2755 ;       VECTORS
2756 ;
2757         *=$E450
2758 ;
2759         JMP     DINIT       ;DISK INIT. VECTOR
2760         JMP     DSKIF       ;DISK INTERFACE ENTRY POINT
2761 ;
2762 ;
2763 ;
2764 ;
2765 ;
2766 ;
2767 ;       CONSTANTS
2768 ;
2769         *=DSKORG
```

```
2770 ;
2771 ;
2772 ;
2773 ;
2774 ;
2775 ;
2776 ;
2777 ;
2778 ;
2779 ;*********************************************************************
2780 ;       DISK INTERFACE ROUTINE STARTS HERE
2781 ;*********************************************************************
2782 ;
2783 ;
2784 ;
2785 ;
2786 ;       DISK INTERFACE INITIALIZATION ROUTINE
2787 ;
2788 DINIT:  LDA     #160
2789         STA     DSKTIM      ;SET INITIAL DISK TIMEOUT TO 160 SEC
2790         RTS
2791 ;
2792 ;
2793 ;
2794 ;       DISK INTERFACE ENTRY POINT
2795 ;
2796 DSKIF:  LDA     #DISKID
2797         STA     DDEVIC      ;SET SERIAL BUS ID IN DCB
2798         LDA     DSKTIM
2799         LDX     DCOMND
2800         CPX     #FOMAT      ;IS COMMAND A FORMAT COMMAND?
2801         BEQ     PUTDTO
2802         LDA     #7          ;NO, SET TIMEOUT TO 7 SECS.
2803 PUTDTO: STA     DTIMLO      ;PUT DISK TIMEOUT IN DCB
2804         LDX     #GETDAT     ;SET "GET DATA" COMMAND FOR SIO
2805         LDY     #$80        ;SET BYTE COUNT TO 128
2806         LDA     DCOMND      ;READ COMMAND IN DCB
2807         CMP     #WRITE      ;IS COMMAND A "PUT SECTOR" COMMAND?
2808         BNE     CKSTC
2809         LDX     #PUTDAT     ;YES, SET "PUT DATA" COMMAND FOR 610
2810 CKSTC:  CMP     #STATC      ;IS COMMAND A STATUS COMMAND?
2811         BNE     PUTCNT
2812         LDA     #STATVL
2813         STA     DBUFLO
2814         LDA     #STATVH
2815         STA     DBUFHI      ;SET BUFFER ADDR TO GLOBAL STATUS BUFFER
2816         LDY     #4          ;YES, SET BYTE COUNT TO 4
2817 PUTCNT: STX     DSTATS      ;PUT STATUS COMMAND FOR SIO IN DCB
2818         STY     DBYTLO
2819         LDA     #0
2820         STA     DBYTHI      ;PUT BYTE COUNT IN DCB
2821         JSR     SIOV        ;CALL SERIAL I/O.
2822         BPL     GOODST      ;NO ERROR
2823         RTS                 ;NO, GO BACK
2824 GOODST: LDA     DCOMND      ;READ THE COMMAND
2825         CMP     #STATC      ;WAS IT A STATUS COMMAND?
2826         BNE     PUTBC
2827         JSR     PUTADR      ;PUT BUFFER ADDR IN TEMP REQ.
2828         LDY     #2
```

```
2829         LDA     (BUFADR),Y  ;READ DISK TIMEOUT VALUE BYTE OF STATUS
2830         STA     DSKTIM      ;PUT IT IN DISK TIMEOUT REQ.
2831 PUTBC:  LDA     DCOMND
2832         CMP     #FOMAT      ;WAS COMMAND A FORMAT COMMAND?
2833         BNE     ENDDIF
2834 FMTD:   JSR     PUTADR      ;YES PUT BUFFER, ADDR INTO TEMP REC
2835         LDY     #$FE        ;SET BUFFER POINTER
2836 TWICE:  INY
2837         INY                 ;INCR BUFFER POINTER BY 2
2838 RDBAD:  LDA     (BUFADR),Y  ;READ LO BYTE BAD SECTOR DATA
2839         CMP     #$FF
2840         BNE     TWICE       ;IS IT "FF" ?
2841         INY                 ;YES,
2842         LDA     (BUFADR),Y  ;READ HI BYTE BAD SECTOR DATA
2843         INY
2844         CMP     #$FF
2845         BNE     RDBAD       ;IS IT "FF" ?
2846         DEY
2847         DEY                 ;YES
2848         STY     DBYTLO      ;PUT BAD SECTOR BYTE COUNT INTO DCB
2849         LDA     #0
2850         STA     DBYTHI
2851 ENDDIF: LDY     DSTATS
2852         RTS
2853 ;
2854 ;
2855 ;
2856 ;
2857 ;       SUBROUTINES
2858 ;
2859 ;
2860 ;       PUT BUFFER ADDR FROM DCB INTO TEMP REQ
2861 ;
2862 PUTADR: LDA     DBUFLO
2863         STA     BUFADR
2864         LDA     DBUFHI
2865         STA     BUFADR+1    ;PUT BUFFER ADDR IN TEMP REQ
2866         RTS
2867 ;**********************************************************************
2868 ;
2869 ;
2870 ;       SPARE BYTE OR MODULE TOO LONG FLAG
2871 ;
2872 CRNTP4  =       *
2873 ;
2874 ;
2875 DSKSPR: .BYTE   PRNORG-CRNTP4 ;^GDISKP TOO LONG
2876 ;
2877         .PAGE
2878         .TITLE 'PRINTER ***** PRINTP.SRC ***** 3/9/79 ***** 4:00:00 P
2879 ;
2880 ;
2881 ;
2882 ;
2883 ;
2884 ;
2885 ;
2886 ;
2887 ;
```

```
2888 ;
2889 ;
2890 ;       DEVICE NUMBER OR CODE EQUATES
2891 ;
2892 OPNOUT  =       $2          ;IOCB OPEN FOR OUTPUT COMMAND
2893 NBUFSZ  =       40          ;PRINT NORMAL BUFFER SIZE
2894 DBUFSZ  =       20          ;PRINT DOUBLE BUFFFER SIZE
2895 SBUFSZ  =       29          ;PRINT SIDEWAYS BUFFER SIZE
2896 PDEVN   =       $40         ;PRINTER DEVICE NUMBER
2897 ; STATC =       $53         ;DCB STATUS COMMAND CODE
2898 WRITEC  =       $57         ;DCB WRITE COMMAND
2899 SPACE   =       $20         ;ASCII SPACE CHAR.
2900 N       =       $4E         ;ASCII "N" CHAR.
2901 D       =       $44         ;ASCII "D" CHAR.
2902 S       =       $53         ;ASCII "S" CHAR.
2903 ;
2904 ;
2905 ;       PRINTER HANDLER ENTRY POINTS
2906 ;
2907 ;
2908 ;
2909 ;
2910         *=$E430
2911 ;
2912         .WORD   PHOPEN-1    ;PRINTER HANDLER OPEN
2913         .WORD   PHCLOS-1    ;PH CLOSE
2914         .WORD   BADST-1     ;PH READ
2915         .WORD   PHWRIT-1    ;PH WRITE
2916         .WORD   PHSTAT-1    ;PH STATUS
2917         .WORD   BADST-1     ;PH SPECIAL
2918         JMP     PHINIT      ;PH INIT.
2919         .BYTE   0           ;ROM FILLER
2920 ;
2921 ;
2922 ;
2923 ;
2924 ;
2925         *=PRNORG
2926 ;
2927 ;
2928 ;
2929 ;
2930 ;       PRINTER HANDLER INITIALIZATION ROUTINE
2931 ;
2932 PHINIT: LDA     #30
2933         STA     PTIMOT      ;SET UP INITIAL PRINTER TIMEOUT OF 30 SEC.
2934         RTS
2935 ;
2936 ;
2937 ;       PRINTER HANDLER CONSTANTS
2938 ;
2939 PHSTLO: .WORD   DVSTAT      ;STATUS BUFFER POINTER
2940 PHCHLO: .WORD   PRNBUF      ;CHAR. BUFFER POINTER
2941 ;
2942 ;
2943 ;
2944 ;       ************************
2945 ;       PRINTER HANDLER ROUTINES
2946 ;       ************************
```

```
2947 ;
2948 ;
2949 ;
2950 ;
2951 ;
2952 ;       PRINTER HANDLER STATUS ROUTINE
2953 ;
2954 PHSTAT: LDA     #4
2955         STA     PBUFSZ      ;SET BUFFER SIZE TO 4 BYTES
2956         LDX     PHSTLO
2957         LDY     PHSTLO+1    ;SET POINTER TO STATUS BUFFER
2958         LDA     #STATC      ;SETCOMMAND TO "STATUS"
2959         STA     DCOMND      ;SET STATUS COMMAND
2960         STA     DAUX1
2961         JSR     SETDCB      ;GO SETUP DCH
2962         JSR     SIOV        ;SEND STATUS COMMAND
2963         BMI     BADST       ;GO IF ERROR
2964         JSR     PHPUT       ;YES, PUT STATUS INTO GLOBAL BUFFER.
2965 BADST:  RTS
2966 ;
2967 ;
2968 ;
2969 ;
2970 ;       PRINTER HANDLER OPEN ROUTINE
2971 ;
2972 PHOPEN: JSR     PHSTAT      ;DO STATUS COMMAND TO SIO
2973         LDA     #0
2974         STA     PBPNT       ;CLEAR PRINT BUFFER POINTER
2975         RTS
2976 ;
2977 ;
2978 ;
2979 ;
2980 ;       PRINTER HANDLER WRITE ROUTINE
2981 ;
2982 PHWRIT: STA     PTEMP       ;SAVE ACCUM
2983         JSR     PRMODE      ;GO DETERMINE PRINTMODE
2984         LDX     PBPNT
2985         LDA     PTEMP       ;GET CHAR. SENT BY CID
2986         STA     PRNBUF,X    ;PUT CHAR. IN PRINT BUFFER
2987         INX                 ;INCR. BUFFER POINTER
2988         CPX     PBUFSZ      ;BUFFER POINTERBUFFER SIZE?
2989         BEQ     BUFFUL
2990         STX     PBPNT       ;SAVE SUFFER POINTER
2991         CMP     #CR         ;IS CHAR. = EOL ?
2992         BEQ     BLFILL      ;IF YES, GO DO BLANK FILL.
2993         LDY     #SUCCES     ;PUT GOOD STATUS IN Y REQ FOR CIO.
2994         RTS
2995 BLFILL: LDA     #SPACE      ;PUT BLANK IN ACCUM.
2996 FILLBF: STA     PRNBUF,X    ;STORE IT IN PRINT BUFFER.
2997         INX
2998         CPX     PBUFSZ
2999         BNE     FILLBF      ;BUFFER BLANK FILLED?
3000 BUFFUL: LDA     #0
3001         STA     PBPNT       ;CLEAR PRINT BUFFER POINTER
3002         LDX     PHCHLO
3003         LDY     PHCHLO+1    ;SET POINTER TO PRINT BUFFER
3004         JSR     SETDCB      ;GO SETUP OCR
3005         JSR     SIOV        ;SEND PRINT COMMAND
```

```
3006         RTS                 ;YES.
3007 ;
3008 ;
3009 ;
3010 ;
3011 ;       PRINTER HANDLER CLOSE ROUTINE
3012 ;
3013 PHCLOS: JSR     PRMODE      ;GO DETERMINE PRINT MODE
3014         LDX     PBPNT
3015         BNE     BLFILL
3016         LDY     #SUCCES
3017         RTS
3018 ;
3019 ;
3020 ;
3021 ;
3022 ;
3023 ;
3024 ;
3025 ;
3026 ;       S U B R O U T I N E S
3027 ;
3028 ;
3029 ;
3030 ;
3031 ;
3032 ;       SET UP DCB TO CALL SIO
3033 ;
3034 SETDCB: STX     DBUFLO
3035         STY     DBUFHI      ;SET BUFFER POINTER
3036         LDA     #PDEVN
3037         STA     DDEVIC      ;SET PRINTER BUS I.D. FOR DCB
3038         LDA     #1
3039         STA     DUNIT       ;SET UNIT NUMBER TO 1
3040         LDA     #$80        ;DEVICE WILL EXPECT DATA
3041         LDX     DCOMND
3042         CPX     #STATC      ;STATUS COMMAND?
3043         BNE     PSIOC
3044         LDA     #$40        ;EXPECT DATA FROM DEVICE
3045 PSIOC:  STA     DSTATS      ;SET SIO MODE COMMAND
3046         LDA     PBUFSZ
3047         STA     DBYTLO      ;SET LO BYTE COUNT
3048         LDA     #0
3049         STA     DBYTHI      ;SET HI BYTE COUNT
3050         LDA     PTIMOT
3051         STA     DTIMLO      ;SET DEVICE TIMEOUT COUNT
3052         RTS
3053 ;
3054 ;
3055 ;
3056 ;
3057 ; GET DEVICE TIMEOUT FROM STATUS & SAVE IT
3058 ;
3059 PHPUT:  LDA     DVSTAT+2
3060         STA     PTIMOT      ;SAVE DEVICE TIMEOUT
3061         RTS
3062 ;
3063 ;
3064 ;
```

```
3065 ;
3066 ; DETERMINE PRINT MODE & SETUP PRINT BUFFER SIZE, DCB PRINT
3067 ; COMMAND, &. DCB AUX1 FOR PRINT MODE
3068 ;
3069 PRMODE: LDY     #WRITEC     ;PUT WRITE COMMAND IN Y REG
3070         LDA     ICAX2Z      ;READ PRINT MODE
3071 CMODE:  CMP     #N
3072         BNE     CDUBL       ;PRINT NORMAL ?
3073         LDX     #NBUFSZ     ;YES, SET NORMAL CHAR. BUFFER SIZE
3074         BNE     SETBSZ
3075 CDUBL:  CMP     #D
3076         BNE     CSIDE       ;PRINT DOUBLE?
3077         LDX     #DBUFSZ     ;YES, SET DOUBLE CHAR. BUFFER SIZE
3078         BNE     SETBSZ
3079 CSIDE:  CMP     #S          ;PRINT SIDEWAYS ?
3080         BNE     GOERR       ;IF NOT, GO TO ERROR ROUTINE
3081         LDX     #SBUFSZ     ;YES, SET SIDEWAYS BUFFER SIZE
3082 SETBSZ: STX     PBUFSZ      ;STORE PRINT BUFFER SIZE
3083         STY     DCOMND      ;STORE DCB COMMAND
3084         STA     DAUX1       ;STORE DCB AUX1 PRINT MODE
3085         RTS
3086 GOERR:  LDA     #N          ;SET DEFAULT PRINT MODE TO NORMAL
3087         BNE     CMODE
3088 ;************************************************************
3089 ;
3090 ;
3091 ;       SPARE BYTE OR MODULE TOO LONG FLAG
3092 ;
3093 CRNTP5  =       *
3094 ;
3095 ;
3096 ;
3097 PRNSPR: .BYTE   CASORG-CRNTP5 ;^GPRINTP TOO LONG
3098 ;
3099         .PAGE
3100         .TITLE  'CASSET HANDLER 3/12 (DK1:CASCV)'
3101 CBUFH   =       CASBUF/256
3102 CBUFL   =       (-256)*CBUFH+CASBUF
3103 SRSTA   =       $40         ;SIO READ STATUS
3104 SWSTA   =       $80         ;SIO WRITE STATUS
3105 ;MOTRGO =       $34
3106 ;MOTRST =       $3C
3107 ;
3108 ;
3109 DTA     =       $FC         ;DATA RECORD TYPE BYTE
3110 DT1     =       $FA         ;LAST DATA RECORD
3111 EOT     =       $FE         ;END OF TAPE
3112 HDR     =       $FB         ;HEADER
3113 TONE1   =       2           ;CHANGE TO RECORD MODE TONE
3114 TONE2   =       1           ;PRESS PLAY TONE
3115 ;
3116 ;
3117 ;
3118         *=CASETV
3119         .WORD   OPENC-1,CLOSEC-1,GBYTE-1,PBYTE-1,STATU-1,SPECIAL-1
3120
3121
3122         JMP     INIT
3123         .BYTE   0           ;ROM FILLER BYTE
```

```
3124 ;
3125 ;
3126 ;
3127 ; USED IN MONITP FOR CASSETTE BOOT
3128 ;
3129         *=RBLOKV
3130         JMP     RBLOK
3131 ;
3132         *=CSOPIV
3133         JMP     OPINP
3134 ;
3135 ;
3136         *=CASORG
3137 ;
3138 ;
3139 ; INIT ROUTINE
3140 ;
3141 INIT:   LDA     #$CC
3142         STA     CBAUDL
3143         LDA     #$05
3144         STA     CBAUDH      ;SET CASSET BAUD RATE TO 600
3145 SPECIAL:                    ;THATS ALL FOLKS
3146         RTS
3147         .PAGE
3148 ;
3149 ; OPEN FUNCTION - WITH NO TIMING ADJUST
3150 ;
3151 OPENC:  LDA     ICAX2Z      ;GET AX2
3152         STA     FTYPE       ;SAVE IT FOR FUTURE REFERENCE
3153         LDA     ICAX1Z
3154         AND     #$0C        ;IN AND OUT BITS
3155         CMP     #$04
3156         BEQ     OPINP
3157         CMP     #$08        ;SEE IF OPEN FOR OUTPUT
3158         BEQ     OPOUT
3159         RTS                 ;IF ALREADY OPEN, RETURN LEAVING STATUS=$84
3160 OPINP:  LDA     #0
3161         STA     WMODE       ;SET READ MODE
3162         STA     FEOF        ;NO EOF YET
3163 SFH:    LDA     #TONE2      ;TONE FOR PRESS PLAY
3164         JSR     BEEP        ;GO BEEP
3165         BMI     OPNRTN      ;IF ERROR DURING BEEP
3166         LDA     #MOTRGO
3167         STA     PACTL       ;TURN MOTOR ON
3168         .IF     PALFLG
3169         LDY     #$E0
3170         LDX     #1
3171         .ENDIF
3172         .IF     PALFLG-1
3173         LDY     #$40        ;5-31-79 9 SEC READ LEADER
3174         LDX     #2
3175         .ENDIF
3176         LDA     #3
3177         STA     CDTMF3
3178         JSR     SETVBV      ;SET UP YBLANK TIMER
3179 WAITTM: LDA     CDTMF3
3180         BNE     WAITTM      ;WAIT FOR MOTOR TO COME UP TO SPEED
3181         LDA     #$80        ;NEXT BYTE=NO BYTES IN BUFFER
3182         STA     BPTR
```

```
3183         STA     BLIM
3184         JMP     OPOK         ;OPEN OK
3185 ;
3186 ; OPEN FOR OUTPUT
3187 ;
3188 PBRK:   LDY     #BRKABT      ;BREAK KEY ABORT STATUS
3189         DEC     BRKKEY       ;RESET BREAK KEY
3190 OPNRTN: LDA     #0           ;CLEAR WRITE MODE FLAG
3191         STA     WMODE
3192         RTS                  ;AND EXIT.
3193 ;
3194 OPOUT:  LDA     #$80
3195         STA     WMODE        ;SET WRITE MODE
3196         LDA     #TONE1       ;TELL USER TO TURN ON RECORD MODE
3197         JSR     BEEP
3198         BMI     OPNRTN       ;IF ERROR DURING BEEP
3199         LDA     #$CC         ;SET BAUD RATE
3200         STA     AUDF3        ;WHICH SEEMS TO BE NESSECARY
3201         LDA     #$05         ;FOR SOME OBSCURE REASON
3202         STA     AUDF4
3203         LDA     #$60
3204         STA     DDEVIC
3205         JSR     SENDEV       ;TELL POKEY TO WRITE MARKS
3206         LDA     #MOTRGO      ;WRITE 5 SEC BLANK TAPE
3207         STA     PACTL
3208         LDA     #3
3209         .IF     PALFLG
3210         LDX     #$3
3211         LDY     #$C0
3212         .ENDIF
3213         .IF     PALFLG-1
3214         LDX     #4           ;5/30/79 20 SEC LEADER
3215         LDY     #$80
3216         .ENDIF
3217         JSR     SETVBV
3218         LDA     #$FF
3219         STA     CDTMF3
3220 WDLR:   LDA     BRKKEY
3221         BEQ     PBRK         ;IF BREAK DURING WRITE LEADER
3222         LDA     CDTMF3
3223         BNE     WDLR
3224         LDA     #0           ;INIT BUFFER POINTER
3225         STA     BPTR
3226 OPOK:   LDY     #SUCCES
3227         RTS
3228         .PAGE
3229 ;
3230 ; GET BYTE
3231 ;
3232 GBYTE:  LDA     FEOF         ;IF AT EOF ALREADY
3233         BMI     ISEOF        ;RETURN EOF STATUS
3234         LDX     BPTR         ;BUFFER POINTER
3235         CPX     BLIM         ;IF END OF BUFFER
3236         BEQ     RBLOK        ;READ ANOTHER BLOCK
3237         LDA     CASBUF+3,X   ;GET NEXT BYTE
3238         INC     BPTR         ;DUMP POINTER
3239         LDY     #SUCCES      ;OK STATUS
3240 GBX:    RTS
3241 RBLOK:  LDA     #'R          ;READ OPCODE
```

```
3242         JSR     SIOSB       ;SIO ON SYS BUF
3243         TYA
3244         BMI     GBX         ;IF SIO ERRORS, RETURN
3245         LDA     #0
3246         STA     BPTR        ;RESET POINTER
3247         LDX     #$80        ;DEFAULT # BYTES
3248         LDA     CASBUF+2
3249         CMP     #EOT
3250         BEQ     ATEOF       ;IF HEADER, GO READ AGAIN
3251         CMP     #DT1        ;IF LAST DATA REC
3252         BNE     NLR
3253         LDX     CASBUF+130  ;LAST DATA RECORD, GET # BYTES
3254 NLR:    STX     BLIM
3255         JMP     GBYTE       ;GET NEXT BYTE
3256 ATEOF:  DEC     FEOF        ;SET FEOF
3257 ISEOF:  LDY     #EOFERR     ;ENDFILE STATUS
3258         RTS
3259         .PAGE
3260 ;
3261 ; PUT BYTE TO BUFFER
3262 ;
3263 PBYTE:  LDX     BPTR        ;BUFFER POINTER
3264         STA     CASBUF+3,X  ;STORE CHAR AWAY
3265         INC     BPTR        ;BUMP POINTER
3266         LDY     #SUCCES     ;OK STATUS
3267         CPX     #127        ;IF BUFFER FULL
3268         BEQ     *+3
3269         RTS
3270 ; WRITE OUT THE BUFFER
3271         LDA     #DTA        ;RECORD TYPE = DATA
3272         JSR     WSIOSB      ;DO WRITE ON SYSTEM BUFFER
3273         LDA     #0
3274         STA     BPTR        ;RESET BUFFER POINTER
3275         RTS                 ;EXIT.
3276         .PAGE
3277 ;
3278 ; STATUS - RETURN STATUS INFO THRU DVSTAT
3279 ;
3280 STATU:  LDY     #SUCCES
3281         RTS
3282         .PAGE
3283 ;
3284 ; CLOSE
3285 ;
3286 CLOSEC: LDA     WMODE       ;SEE IF WRITING
3287         BMI     CLWRT       ;300 CLOSE FOR WRITE
3288 ; CLOSE FOR READ - FLAG CLOSED
3289         LDY     #SUCCES     ;SUCCESSFULL
3290 FCAX:   LDA     #MOTRST     ;STOP THE MOTOR IN CASE WAS SHORT IRQ MODE
3291         STA     PACTL
3292         RTS
3293 CLWRT:  LDX     BPTR        ;BUFFER POINTER
3294         BEQ     WTLR        ;IF NO DATA BYTES IN BUFFER, NO DT1 REC
3295         STX     CASBUF+130  ;WRITE TO LAST RECORD
3296         LDA     #DT1        ;REC TYPE
3297         JSR     WSIOSB      ;WRITE OUT USER BUFFER
3298         BMI     FCAX        ;GO IF ERROR
3299 WTLR:   LDX     #127        ;ZERO BUFFER
3300         LDA     #0
```

```
3301 ZTBUF:  STA     CASBUF+3,X
3302         DEX
3303         BPL     ZTBUF
3304         LDA     #EOT        ;WRITE EOT RECORD
3305         JSR     WSIOSB
3306         JMP     FCAX        ;FLAG CLOSED AND EXIT
3307         .PAGE
3308 ;
3309 ; SUBROUTINES
3310 ;
3311 ; BEEP - GENERATE TONE ON KEYBOARD SPEAKER
3312 ; ON ENTRY A= FREQ
3313 ;
3314 BEEP:   STA     FREQ
3315 BEEP1:  LDA     RTCLOK+2    ;CURRENT CLOCK
3316         CLC
3317         .IF     PALFLG
3318         ADC     #25
3319         .ENDIF
3320         .IF     PALFLG-1
3321         ADC     #30         ; 1 SEC TONE
3322         .ENDIF
3323         TAX
3324 WFL:    LDA     #$FF
3325         STA     CONSOL      ;TURN ONSPEAKER
3326         LDA     #0
3327         LDY     #$F0
3328         DEY
3329         BNE     *-1
3330         STA     CONSOL      ;TURN OFF SPEAKER
3331         LDY     #$F0
3332         DEY
3333         BNE     *-1
3334         CPX     RTCLOK+2    ;SEE IF 1 SEC IS UP YET
3335         BNE     WFL
3336         DEC     FREQ        ;COUNT BEEPS
3337         BEQ     WFAK        ;IF ALL DONE GO WAIT FOR KEY
3338         TXA
3339         CLC
3340         .IF     PALFLG
3341         ADC     #8
3342         .ENDIF
3343         .IF     PALFLG-1
3344         ADC     #10
3345         .ENDIF
3346         TAX
3347         CPX     RTCLOK+2
3348         BNE     *-2
3349         BEQ     BEEP1       ;UNCOND DO BEEP AGIN
3350 WEAK:   JSR     WFAK1       ;USE SIMULATED "JMP (KGETCH)"
3351         TYA
3352         RTS
3353 WFAK1:  LDA     KEYBDV+5
3354         PHA
3355         LDA     KEYBDV+4    ;SIMULATE "JMP (KGETCH)"
3356         PHA
3357         RTS
3358 ;
3359 ; SIOBS - CALL SIO ON SYSTEM BUFFER
```

```
3360 ;
3361 SIOSB:  STA     DCOMND      ;SAVE COMMAND
3362         LDA     #0
3363         STA     DBYTHI      ;SET BUFFER LENGTH
3364         LDA     #131
3365         STA     DBYTLO
3366         LDA     #CBUFH
3367         STA     DBUFHI      ;SET BUFFER ADDRESS
3368         LDA     #CBUFL
3369         STA     DBUFLO
3370 CSIO:   LDA     #$60        ;CASSET PSEUDO DEVICE
3371         STA     DDEVIC
3372         LDA     #0
3373         STA     DUNIT
3374         LDA     #35         ;DEVICE TIMEOUT (5/30/79)
3375         STA     DTIMLO
3376         LDA     DCOMND      ;GET COMMAND SACK
3377         LDY     #SRSTA      ;SIO READ STATUS COMMAND
3378         CMP     #'R
3379         BEQ     *+4
3380         LDY     #SWSTA      ;SIC WRITE STATUS COMMAND
3381         STY     DSTATS      ;SET STATUS FOR SIO
3382         LDA     FTYPE
3383         STA     DAUX2       ;INDICATE IF SHORT IRQ MODE
3384         JSR     SIOV        ;GO CALL SIO
3385         RTS
3386 ;
3387 ; WSIOSB - WRITE SIC SYSTEM SUFFER
3388 ;
3389 WSIOSB: STA     CASBUF+2    ;STORE TYPE BYTE
3390         LDA     #$55
3391         STA     CASBUF+0
3392         STA     CASBUF+1
3393         LDA     #'W         ;WRITE
3394         JSR     SIOSB       ;CALL SIO ON SYSTEM BUFFER
3395         RTS     AND         ;RETURN
3396 CRNTP6  =*
3397         *=$14
3398 CASSPR: .BYTE   MONORG-CRNTP6 ;^GCASCV IS TOO LONG
3399 ;
3400         .TITLE  'MONITOR  ***** MONITP.SRC ***** 3/9/79 *****  4:00:00 P
3401 ;
3402 ;
3403 ;
3404 ;       CONSTANT EQUATES
3405 ;
3406 PUTTXT  =       $9            ;"PUT TEXT RECORD" CIO COMMANDCODE
3407 GETCAR  =       $7            ;"GET CHARACTER" CIO COMMAND CODE
3408 PUTCAR  =       $B            ;"PUT CHARACTER" CIO COMMAND CODE
3409 INIMLL  =       $00           ;INITIAL HEM LO LOW BYTE
3410 INIMLH  =       $07           ;INITIAL HEM LO HIGH BYTE
3411 ; GOOD  =       $1            ;GOOD STATUS CODE
3412 ; WRITE =       $57           ;WRITE COMMAND
3413 ; READ  =       $52           ;READ COMMAND
3414 ; STATC =       $53           ;STATUS COMMAND
3415 SEX     =       $0            ;SCREEN EDITOR 10CR INDEX
3416 CLS     =       $7D           ;CLEAR SCREEN CODE
3417 CTRLC   =       $92           ;KEYBOARD CODE FOR 'CONTROL C'
3418 EOF     =       $88           ;CASSETTE END OF FILE CODE
```

```
3419 LIRQ    =       $0             ;LONG IRQ TYPE CODE
3420 ;
3421 BUFFH   =       (CASBUF+3)/256
3422 BUFFL   =       (-256)*BUFFH+CASBUF+3 ;BUFFER POINTER
3423 ;
3424 ;
3425 ;
3426 ; THE FOLLOWING EQUATES ARE IN THE CARTRIDGE ADDRESS SPACE.
3427 ;
3428 ;
3429 ; "B" CARTRIDGE ADDR'S ARE 8000-9FFF (36K CONFIG. ONLY)
3430 ; "A" CART. ADDR'S ARE A000-BFFF (36K CONFIG. ONLY)
3431 ;
3432 ; "A" CART. ADDR'S ARE B000-BFFF (48K CONFIG. ONLY)
3433 ;
3434         *=$BFFA
3435 CARTCS: .RES    2             ;CARTRIDGE COLD START ADDRESS.
3436 CART:   .RES    1             ;CARTRIDGE AVAILABLE FLAG BYTE.
3437 CARTFG: .RES    1             ;CARTRIDGE FLAG BYTE. BIT 0=FLAG1,
3438 CARTAD: .RES    2             ;2-BYTE CARTRIDGE START VECTOR
3439 ;
3440 ;
3441 ;       CARTRIDGE FLAG ACTION DEFINITIONS
3442 ;
3443 ;
3444 ;       BIT             ACTION IF SET
3445 ;
3446 ;       7               SPECIAL -- DON'T POWER-UP, JUST RUN CARTRIDGE
3447 ;       6-3             NONE
3448 ;       2               RUN CARTRIDGE
3449 ;       1               NONE
3450 ;       0               BOOT DOS
3451 ;
3452 ;
3453 ;       *****
3454 ;       NOTE
3455 ;       *****
3456 ;
3457 ;       1.IF BIT2 IS 0, GOTO BLACKBOARD MODE.
3458 ;       2.IF BIT0 SET THE DISK WILL BE BOOTED BEFORE ANY
3459 ;         OTHER ACTION.
3460 ;
3461 ;
3462 ;
3463 ;
3464 ;
3465 ;
3466 ;
3467 ;
3468 ;
3469 ;       POWER-UP VECTOR
3470 ;
3471 ;***************************
3472 ;       *=$FFFC
3473 ;
3474 ; PVECT .WORD   PWRUP           POWER-UP VECTOR
3475 ;************************
3476 ;
3477 ;
```

```
3478 ;
3479 ;
3480 ;
3481 ;       ENTRY POINT VECTOR
3482 ;
3483         *=BLKBDV
3484 ;
3485         JMP     SIGNON      ;BLACK BOARD VECTOR
3486 ;
3487         *=WARMSV
3488 ;
3489         JMP     RESET       ;WARM START VECTOR
3490 ;
3491         *=COLDSV
3492 ;
3493         JMP     PWRUP       ;COLD START VECTOR (9000 FOR RAM VECTOR WRIT
3494 ;
3495         *=$9000
3496         JSR     $900C
3497         JMP     PWRUP       ;(TO HANDLE RAM VECTOR WRITING)
3498         JSR     $900C
3499         JMP     RESET
3500 ;
3501 ;
3502 ;
3503         *=MONORG
3504 ;
3505 ;
3506 ;
3507 ;
3508 ;       HANDLER TABLE ENTRIES
3509 ;
3510 TBLENT: .BYTE   'P'
3511         .WORD   PRINTV
3512         .BYTE   'C'
3513         .WORD   CASETV
3514         .BYTE   'E'
3515         .WORD   EDITRV
3516         .BYTE   'S'
3517         .WORD   SCRENV
3518         .BYTE   'K'
3519         .WORD   KEYBDV
3520 ;
3521 ;
3522 ;TBLLEN  =       IDENT-TBLENT-1  HANDLER TABLE LENGTH.  "MOVED TO LINE 8
3523 ;
3524 ;       ***** PRINT MESSAGES *****
3525 ;
3526 ;
3527 IDENT:  .BYTE   CLS,'ATARI COMPUTER - MEMO PAD',CR
3528
3529
3530
3531
3532
3533
3534 ;
3535 IDENTH  =       IDENT/256
3536 IDENTL  =       (-256)*IDENTH+IDENT ;SYSTEM I.D. MSG POINTER
```

```
3537 ;
3538 TBLLEN  =       IDENT-TBLENT-1 ;HANDLER TABLE LENGTH
3539 DERR5:  .BYTE   'BOOT ERROR',CR
3540
3541
3542 ;
3543 DERRH   =       DERR5/256
3544 DERRL   =       (-256)*DERRH+DERR5 ;DISK ERROR MSG POINTER
3545 ;
3546 ;
3547 ;
3548 ;
3549 ;       DEVICE/FILENAME SPECIFICATIONS
3550 ;
3551 OPNEDT: .BYTE   'E:',CR ;"OPEN SCREEN EDITOR" DEVICE SPEC.
3552 ;
3553 OPNH    =       OPNEDT/256
3554 OPNL    =       (-256)*OPNH+OPNEDT ;SCREEN EDITOR OPEN POINTER
3555
3556 ;
3557 ;
3558 ;
3559 ;
3560 ;****************************************************************
3561 ;       RESET BUTTON ROUTINE STARTS HERE
3562 ;
3563 ;
3564 RESET:  SEI                     ;DISABLE IRQ INTERRUPTS
3565         LDA     COLDST          ;WERE WE IN MIDDLE OF COLDSTART?
3566         BNE     PWRUP           ;YES, GO TRY IT AGAIN
3567         LDA     #$FF
3568         BNE     PWRUP1          ;SET WARM START FLAG
3569 ;
3570 ;
3571 ;
3572 ;**************************************************************
3573 ;       POWER UP ROUTINES START HERE
3574 ;**************************************************************
3575 ;
3576 PWRUP:  SEI                     ;DISABLE IRQ INTERRUPTS
3577         LDA     #0              ;CLEAR WARMSTART FLAG
3578 PWRUP1: STA     WARMST
3579         CLD                     ;CLEAR DECIMAL FLAG.
3580         LDX     #$FF
3581         TXS                     ;SET STACK POINTER
3582         JSR     SPECL           ;CARTRIDGE SPECIAL CASE?
3583         JSR     HARDI           ;DO HARDWARE INITIALIZATION
3584         LDA     WARMST          ;IS IT WARMSTART?
3585         BNE     ZOSRAM          ;YES, ONLY ZERO OS RAM
3586 ;
3587 ZERORM: LDA     #0
3588         LDY     #WARMST
3589         STA     RAMLO
3590         STA     RAMLO+1         ;INITIALIZE RAM POINTER
3591 CLRRAM: STA     (RAMLO),Y       ;CLEAR MEMORY LOC.
3592         INY
3593         CPY     #0              ;AT END OF PAGE?
3594         BNE     CLRRAM
3595         INC     RAMLO+1         ;YES. INCR PAGE POINTER
```

```
3596         LDX     RAMLO+1
3597         CPX     TRAMSZ      ;AT END OF MEM?
3598         BNE     CLRRAM      ;NO.
3599 ;
3600 ; INITIALIZE DOSVEC TO POINT TO SIGNON (BLACKBOARD)
3601         LDA     BLKBDV+1
3602         STA     DOSVEC      ;USE BLACKBOARD VECTOR
3603         LDA     BLKBDV+2    ;FOR DOSVEC
3604         STA     DOSVEC+1
3605         LDA     #$FF
3606         STA     COLDST      ;SET TO SHOW IN MIDDLE OF COLDSTART
3607         BNE     ESTSCM      ;GO AROUND ZOSRAM
3608 ;
3609 ; CLEAR OS RAM (FOR WARMSTART)
3610 ZOSRAM: LDX     #0
3611         TXA
3612 ZOSRM2: STA     $200,X      ;CLEAR PAGES 2 AND 3
3613         STA     $300,X
3614         DEX
3615         BNE     ZOSRM2
3616         LDX     #INTZBS
3617 ZOSRM3: STA     0,X         ;CLEAR ZERO PAGE LOCATIONS INTZBS-7F
3618         INX
3619         BPL     ZOSRM3
3620 ;
3621 ; ESTABLISH SCREEN MARGINS
3622 ESTSCM: LDA     #LEDGE
3623         STA     LMARGN
3624         LDA     #REDGE
3625         STA     RMARGN
3626 ;
3627 ;
3628 ; MOVE VECTOR TABLE FROM ROM TO RAM
3629 OPSYS:  LDX     #$25
3630 MOVVEC: LDA     VCTABL,X    ;ROM TABLE
3631         STA     INTABS,X    ;TO RAM
3632         DEX
3633         BPL     MOVVEC
3634         JSR     OSRAM       ;DO O.S. RAM SETUP
3635         CLI                 ;ENABLE IRQ INTERRUPTS
3636 ;
3637 ;
3638 ;       LINK HANDLERS
3639 ;
3640         LDX     #TBLLEN
3641 NXTENT: LDA     TBLENT,X    ;READ HANDLER TABLE ENTRY
3642         STA     HATABS,X    ;PUT IN TABLE
3643         DEX
3644         BPL     NXTENT      ;DONE WITH ALL ENTRIES?
3645 ;
3646 ;
3647 ;
3648 ;
3649 ;
3650 ; INTERROGATE CARTRIDGE ADDR. SPACE TO SEE WHICH CARTRIDGES THERE ARE
3651 ;
3652         LDX     #0
3653         STX     TSTDAT      ;CLEAR "B" CART. FLAG
3654         STX     TRAMSZ      ;CLEAR "A" CART. FLAG
```

```
3655         LDX     RAMSIZ
3656         CPX     #$90        ;RAM IN "B" CART. SLOT?
3657         BCS     ENDBCK
3658         LDA     CART-$2000  ;NO.
3659         BNE     ENDBCK      ;CART. PLUGGED INTO "B" SLOT'?
3660         INC     TSTDAT      ;YES, SET "B" CART, FLAG
3661         JSR     CBINI       ;INITIALIZE CARTRIDGE "B"
3662 ;
3663 ENDBCK: LDX     RAMSIZ
3664         CPX     #$B0        ;RAM IN "A" CART. SLOT?
3665         BCS     ENDACK
3666         LDX     CART        ;NO,
3667         BNE     ENDACK      ;CART. PLUGGED INTO "A" SLOT?
3668         INC     TRAMSZ      ;YES, SET "A" CART. FLAG
3669         JSR     CAINI       ;INITIALIZE CARTRIDGE "A"
3670 ;
3671 ;
3672 ; OPEN SCREEN EDITOR
3673 ;
3674 ENDACK: LDA     #3
3675         LDX     #SEX
3676         STA     ICCOM,X      ;OPEN I/O COMMAND
3677         LDA     #OPNL
3678         STA     ICBAL,X
3679         LDA     #OPNH
3680         STA     ICBAH,X     ;SET BUFFER POINTER TO OPEN SCREEN EDITOR
3681         LDA     #$C
3682         STA     ICAX1,X     ;SET UP OPEN FOR INPUT/OUTPUT
3683         JSR     CIOV        ;GO TO CIO
3684 ;
3685         BPL     SCRNOK      ;BR IF NO ERROR
3686         JMP     PWRUP       ;RETRY PWRUP IF ERROR (SHOULD NEVER HAPPEN!)
3687 SCRNOK: INX                 ;SCREEN OK, SO WAIT FOR YBLANK TO
3688         BNE     SCRNOK      ;BRING UP THE DISPLAY
3689         INY
3690         BPL     SCRNOK
3691 ;
3692 ;
3693 ; DO CASSETTE BOOT
3694         JSR     CSBOOT      ;CHECK, BOOT, AND INIT
3695 ;
3696 ; CHECK TO SEE IF EITHER CARTRIDGE WANTS DISK BOOT
3697         LDA     TRAMSZ      ;CHECK BOTH CARTRIDGES
3698         ORA     TSTDAT      ;
3699         BEQ     NOCART      ;NEITHER CARTRIDGE LIVES
3700         LDA     TRAMSZ      ;"A" CART?
3701         BEQ     NOA1        ;NO
3702         LDA     CARTFG      ;GET CARTRIDGE MODE FLAG
3703 NOA1:   LDX     TSTDAT      ;"B" CART?
3704         BEQ     NOB1        ;NO
3705         ORA     CARTFG-$2000 ;ADD OTHER FLAG
3706 NOB1:   AND     #1          ;DOES EITHER CART WANT BOOT?
3707         BEQ     NOBOOT      ;NO
3708 ;
3709 ; DO DISK BOOT
3710 NOCART: JSR     BOOT        ;CHECK. BOOT. AND INIT
3711 ;
3712 ; GO TO ONE OF THE CARTRIDGES IF THEY SO DESIRE
3713 NOBOOT: LDA     #0
```

```
3714         STA     COLDST      ;RESET TO SHOW DONE WITH COLDSTART
3715         LDA     TRAMSZ      ;"A" CART?
3716         BEQ     NOA2        ;NO
3717         LDA     CARTFG      ;GET CARTRIDGE MODE FLAG
3718         AND     #4          ;DOES IT WANT TO RUN?
3719         BEQ     NOA2        ;NO
3720         JMP     (CARTCS)    ;RUN "A" CARTRIDGE
3721 NOA2:   LDA     TSTDAT      ;"B" CART?
3722         BEQ     NOCAR2      ;NO
3723         LDA     CARTFG-$2000 ;GET "B" MODE FLAG
3724         AND     #4          ;DOES IT WANT TO RUN?
3725         BEQ     NOCART      ;NO
3726         JMP     (CARTCS-$2000) ;RUN "B" CARTRIDGE
3727 ;
3728 ; NO CARTRIDGES, OR NEITHER WANTS TO RUNS
3729 ; SO GO TO DOSVEC (DOS, CASSETTE, OR BLACKBOARD)
3730 NOCAR2: JMP     (DOSVEC)
3731 ;
3732 ; PRINT SIGN-ON MESSAGE
3733 SIGNON: LDX     #IDENTL
3734         LDY     #IDENTH
3735         JSR     PUTLIN      ;GO PUT SIGN-ON MSG ON SCREEN
3736 ;
3737 ;
3738 ;
3739 ;       BLACKBOARD ROUTINE
3740 BLACKB: JSR     BLKB2       ;"JSR EGETCH"
3741         JMP     BLACKB      ;FOREVER
3742 BLKB2:  LDA     EDITRV+5    ;HIGH BYTE
3743         PHA
3744         LDA     EDITRV+4    ;LOW BYTE
3745         PHA
3746         RTS                 ;SIMULATES "JMP (EDITRV)"
3747 ;
3748 ;
3749 ; CARTRIDGE INITIALIZATION INDIRECT JUMPS
3750 CAINI:  JMP     (CARTAD)
3751 CBINI:  JMP     (CARTAD-$2000)
3752         .PAGE
3753 ;
3754 ;
3755 ;
3756 ;
3757 ;
3758 ;
3759 ;               S U B R O U T I N E S
3760 ;
3761 ;
3762 ;
3763 ;
3764 ;
3765 ;
3766 ;
3767 ;
3768 ;
3769 ;
3770 ;
3771 ;
3772 ;
```

```
3773 ;
3774 ;
3775 ;
3776 ;
3777 ;
3778 ; CHECK FOR HOW MUCH RAM & SPECIAL CARTRIDGE CASE.
3779 ; IF SPECIAL CARTRIDGE CASE, DON'T GO BACK -- GO TO CART.
3780 ;
3781 SPECL:  LDA     CART        ;CHECK FOR RAM OR CART
3782         BNE     ENSPE2      ;GO IF NOTHING OR MAYBE RAM
3783         INC     CART        ;NOW DO RAM CHECK
3784         LDA     CART        ;IS IT ROM?
3785         BNE     ENSPEC      ;NO
3786         LDA     CARTFG      ;YES,
3787         BPL     ENSPEC      ;BIT SET?
3788         JMP     (CARTAD)    ;YES, GO RUN CARTRIDGE
3789 ;
3790 ; CHECK FOR AMOUNT OF RAM
3791 ;
3792 ;
3793 ENSPEC: DEC     CART        ;RESTORE RAM IF NEEDED
3794 ENSPE2: LDY     #0
3795         STY     RAMLO+1
3796         LDA     #$10
3797         STA     TRAMSZ      ;SET RAM POINTER TO 4K.
3798 HOWMCH: LDA     (RAMLO+1),Y ;READ RAM LOCATION
3799         EOR     #$FF        ;INVERT IT.
3800         STA     (RAMLO+1),Y ;WRITE INVERTED DATA.
3801         CMP     (RAMLO+1),Y ;READ RAM AGAIN
3802         BNE     ENDRAM
3803         EOR     #$FF        ;CONVERT IT BACK
3804         STA     (RAMLO+1),Y ;RESTORE ORIGINAL RAMDATA
3805         LDA     TRAMSZ
3806         CLC
3807         ADC     #$10
3808         STA     TRAMSZ      ;INCR. RAM POINTER BY 4K.
3809         BNE     HOWMCH      ;GO FIND HOW MUCH RAM.
3810 ENDRAM: RTS
3811
3812 ;
3813 ;
3814 ;
3815 ;       HARDWARE INITIALIZATION
3816 ;
3817 ;
3818 HARDI:  LDA     #0
3819         TAX
3820 CLRCHP: STA     $D000,X
3821         STA     $D400,X
3822         STA     $D200,X
3823         STA     $D300,X
3824         INX
3825         BNE     CLRCHP
3826         RTS
3827 ;
3828 ;
3829 ;       O.S. RAM SETUP
3830 ;
3831 OSRAM:  DEC     BRKKEY      ;TURN OFF BREAK KEY FLAG
```

```
3832         LDA     #.LOW.BRKKY2
3833         STA     BRKKY
3834         LDA     #.HIGH.BRKKY2
3835         STA     BRKKY+1
3836         LDA     TRAMSZ       ;READ RAM SIZE IN TEMP. REG.
3837         STA     RAMSIZ       ;SAVE IT IN RAM SIZE.
3838         STA     MEMTOP+1     ;INIT. MEMTOP ADDR HI BYTE
3839         LDA     #0
3840         STA     MEMTOP       ;INIT. MEMTOP ADDR LO BYTE
3841         LDA     #INIMLL
3842         STA     MEMLO
3843         LDA     #INIMLH
3844         STA     MEMLO+1      ;INITIALIZE MEMLO ADDR VECTOR
3845         JSR     EDITRV+$C    ;EDITOR INIT.
3846         JSR     SCRENV+$C    ;SCREEN INIT.
3847         JSR     KEYBDV+$C    ;KEYBOARD INIT.
3848         JSR     PRINTV+$C    ;PRINTER HANDLER INIT
3849         JSR     CASETV+$C    ;CASSETTE HANDLER INIT
3850         JSR     CIOINV       ;CIO INIT.
3851         JSR     SIOINV       ;SIO INIT.
3852         JSR     INTINV       ;INTERRUPT HANDLER INIT.
3853         LDA     CONSOL
3854         AND     #$1
3855         BNE     NOKEY        ;GAME START KEY DEPRESSED?
3856         INC     CKEY         ;YES. SET KEY FLAG.
3857 NOKEY:  RTS
3858 ;
3859 ;
3860 ; DO BOOT OF DISK
3861 ;
3862 BOOT:   LDA     WARMST
3863         BEQ     NOWARM       ;WARMSTART?
3864         LDA     BOOT?        ;YES,
3865         AND     #1
3866         BEQ     NOINIT       ;VALID BOOT?
3867         JSR     DINI         ;YES, RE-INIT. DOS SOFTWARE
3868 NOINIT: RTS
3869 NOWARM: LDA     #1
3870         STA     DUNIT        ;ASSIGN DISK DRIVE NO.
3871         LDA     #STATC
3872         STA     DCOMND       ;SET UPSTATUS COMMAND
3873         JSR     DSKINV       ;GO DO DISK STATUS
3874         BPL     DOBOOT       ;IS STATUS FROM 510 GOOD?
3875         RTS                  ;NO, GO BACK WITH BAD BOOT STATUS
3876 ;
3877 DOBOOT: LDA     #0
3878         STA     DAUX2
3879         LDA     #1
3880         STA     DAUX1        ;SET SECTOR # TO 1.
3881         LDA     #BUFFL
3882         STA     DBUFLO
3883         LDA     #BUFFH
3884         STA     DBUFHI       ;SET UP BUFFER ADDR
3885 SECT1:  JSR     GETSEC       ;GET SECTOR
3886         BPL     ALLSEC       ;STATUS O.K.?
3887 BADDSK: JSR     DSKRDE       ;NO, GO PRINT DISK READ ERROR
3888         LDA     CASSBT
3889         BEQ     DOBOOT       ;CASSETTE BOOT?
3890         RTS                  ;YES, QUIT
```

```
3891 ALLSEC: LDX     #3
3892 RDBYTE: LDA     CASBUF+3,X  ;READ A BUFFER BYTE
3893         STA     DFLAGS,X    ;STORE IT
3894         DEX
3895         BPL     RDBYTE      ;DONE WITH 4 BYTE TRANSFER
3896         LDA     BOOTAD      ;YES.
3897         STA     RAMLO
3898         LDA     BOOTAD+1
3899         STA     RAMLO+1     ;PUT BOOT ADDR INTO Z. PAGE RAM
3900         LDA     CASBUF+7
3901         STA     DOSINI      ;ESTABLISH DOS INIT ADDRESS
3902         LDA     CASBUF+8
3903         STA     DOSINI+1
3904 MVBUFF: LDY     #$7F        ;YES, SET BYTE COUNT
3905 MVNXB:  LDA     CASBUF+3,Y
3906         STA     (RAMLO),Y   ;MOVE A BYTE FROM SECTOR BUFFER TO BOOT ADDR
3907         DEY
3908         BPL     MVNXB       ;DONE ?
3909         CLC                 ;YES,
3910         LDA     RAMLO
3911         ADC     #$80
3912         STA     RAMLO
3913         LDA     RAMLO+1
3914         ADC     #0
3915         STA     RAMLO+1     ;INCR BOOT LOADER BUFFER POINTER
3916         DEC     DBSECT      ;DECR # OF SECTORS.
3917         BEQ     ENBOOT      ;MORE SECTORS ?
3918         INC     DAUX1       ;YES INCR SECTOR #
3919 SECTX:  JSR     GETSEC      ;GO GET SECTOR.
3920         BPL     MVBUFF      ;STATUS O.K. ?
3921         JSR     DSKRDE      ;NO, GO PRINT DISK READ ERROR
3922         LDA     CASSBT
3923         BNE     BADDSK      ;IF CASSETTE, QUIT.
3924         BEQ     SECTX       ;IF DISK, TRY SECTOR AGAIN.
3925 ENBOOT: LDA     CASSBT
3926         BEQ     XBOOT       ;A CASSETTE BOOT ?
3927         JSR     GETSEC      ;YES, GET EOF RECORD, BUT DON'T USE IT.
3928 XBOOT:  JSR     BLOAD       ;GO EXECUVE BOOT LOADER
3929         BCS     BADDSK      ;IF BAD BOOT, DO IT OVER AGAIN
3930         JSR     DINI        ;GO INIT. SOFTWARE
3931         INC     BOOT?       ;SHOW BOOT SUCCESS
3932         RTS
3933 BLOAD:  CLC
3934         LDA     BOOTAD
3935         ADC     #6
3936         STA     RAMLO
3937         LDA     BOOTAD+1
3938         ADC     #0
3939         STA     RAMLO+1     ;PUT START ADDR OF BOOTLOADER INTO RAM
3940         JMP     (RAMLO)
3941 DINI:   JMP     (DOSINI)
3942 ;
3943 ;
3944 ;
3945 ;
3946 ; DISPLAY DISK READ ERROR MSG
3947 ;
3948 DSKRDE: LDX     #DERRL
3949         LDY     #DERRH
```

```
3950 ;
3951 ;
3952 ;
3953 ; PUT LINE ON SCREEN AT PRESENT CURSOR POSITION
3954 ;
3955 ;   X-REG -- LO BYTE, BEGIN ADDR OF LINE
3956 ;   Y-REG -- HI BYTE, BEGIN ADDR OF LINE
3957 ;
3958 PUTLIN: TXA
3959         LDX     #SEX
3960         STA     ICBAL,X
3961         TYA
3962         STA     ICBAH,X     ;SET UP ADDR OF BEGIN OF LINE
3963         LDA     #PUTTXT
3964         STA     ICCOM,X     ;"PUT TEXT RECORD" COMMAND
3965         LDA     #$FF
3966         STA     ICBLL,X     ;SET BUFFER LENGTH
3967         JSR     CIOV        ;PUT LINE ON SCREEN
3968         RTS
3969 ;
3970 ;
3971 ;
3972 ;
3973 ; GET SECTOR FROM DISK 0
3974 ;
3975 GETSEC: LDA     CASSBT
3976         BEQ     DISKM       ;CASSETTE BOOT?
3977         JMP     RBLOKV      ;YES, GO TO READ BLOCK ROUTINE
3978 DISKM:  LDA     #READ
3979         STA     DCOMND      ;SET READ SECTOR COMMAND
3980         LDA     #1
3981         STA     DUNIT       ;SET DRIVE NO. TO DRIVE 0
3982         JSR     DSKINV      ;GET SECTOR
3983         RTS
3984 ;
3985 ;
3986 ;
3987 ; DO CHECK FOR CASSETTE BOOT & IF SO DO BOOT
3988 ;
3989 CSBOOT: LDA     WARMST      ;WARMSTART?
3990         BEQ     CSBOT2      ;NO
3991         LDA     BOOT?       ;GET BOOT FLAG
3992         AND     #2          ;WAS CASSETTE BOOT SUCCESFULL?
3993         BEQ     NOCSB2      ;NO
3994         JSR     CINI        ;YES, INIT CASSETTE SOFTWARE
3995 NOCSB2: RTS
3996 ;
3997 CSBOT2: LDA     CKEY
3998         BEQ     NOCSBT      ;"C" KEY FLAG SET ?
3999         LDA     #$80        ;YES,
4000         STA     FTYPE       ;SET LONG IRQ TYPE
4001         INC     CASSBT      ;SET CASSETTE BOOT FLAG
4002         JSR     CSOPIV      ;OPEN CASSETTE FOR INPUT
4003         JSR     SECT1       ;DO BOOT & INIT.
4004         LDA     #0
4005         STA     CASSBT      ;RESET CASSETTE BOOT FLAG
4006         STA     CKEY        ;CLEAR KEY FLAG
4007         ASL     BOOT?       ;SHIFT BOOT FLAG (NOW=2 IF SUCCESS)
4008         LDA     DOSINI
```

```
4009         STA     CASINI      ;MOVE INIT ADDRESS FOR CASSETTE
4010         LDA     DOSINI+1
4011         STA     CASINI+1
4012 NOCSBT: RTS
4013 ;
4014 CINI:   JMP     (CASINI)    ;INIT CASSETTE
4015 ;**************************************************************
4016 ;
4017 ;
4018 ; SPARE BYTE OR MODULE TOO LONG FLAG
4019 ;
4020 CRNTP7  =*
4021 ;
4022         *=$14
4023 MONSPR: .BYTE   KBDORG-CRNTP7 ;^GMONITP TOO LONG
4024 ;
4025         .PAGE
4026         .TITLE  'DISPLAY HANDLER  -- 10-30-78 -- DISPLC'
4027 ;
4028 ; HANDLER DEPENDENT EQUATES
4029 ;
4030 CLRCOD  =       $7D         ;CLEAR SCREEN ATASCI CODE
4031 CNTL1   =       $9F         ;POKEY KEY CODE FOR ^1
4032 ;
4033 FRMADR  =       SAVADR
4034 TOADR   =       MLTTMP
4035 ;
4036         .PAGE
4037 ;
4038 ;
4039         *=EDITRV
4040 ;
4041 ; SCREEN EDITOR HANDLER ENTRY POINT
4042 ;
4043 EDITOR: .WORD   EOPEN-1
4044         .WORD   RETUR1-1    ;(CLOSE)
4045         .WORD   EGETCH-1
4046         .WORD   EOUTCH-1
4047         .WORD   RETUR1-1    ;(STATUS)
4048         .WORD   NOFUNC-1    ;(SPECIAL)
4049         JMP     PWRONA
4050         .BYTE   0           ;ROM FILLER BYTE
4051 ;
4052         *=SCRENV
4053 ;
4054 ; DISPLAY HANDLER ENTRY POINT
4055 ;
4056 DISPLA: .WORD   DOPEN-1
4057         .WORD   RETUR1-1    ;(CLOSE)
4058         .WORD   GETCH-1
4059         .WORD   OUTCH-1
4060         .WORD   RETUR1-1    ;(STATUS)
4061         .WORD   DRAW-1      ;(SPECIAL)
4062         JMP     PWRONA
4063         .BYTE   0           ;ROM FILLER BYTE
4064 ;
4065 ;
4066 ;
4067 ;
```

```
4068 ; KEYBOARD HANDLER ENTRY POINT
4069 ;
4070 KBDHND: .WORD   RETUR1-1
4071         .WORD   RETUR1-1    ;(CLOSE)
4072         .WORD   KGETCH-1
4073         .WORD   NOFUNC-1    ;(OUTCH)
4074         .WORD   RETUR1-1    ;(STATUS)
4075         .WORD   NOFUNC-1    ;(SPECIAL)
4076         JMP     PWRONA
4077         .BYTE   0           ;ROM FILLER BYTE
4078 ;
4079 ;
4080 ; INTERRUPT VECTOR TABLE ENTRY
4081         *=VCTABL-INTABS+VKEYBD
4082         .WORD   PIRQ5       ;KEYBOARD IRQ INTERRUPT VECTOR
4083 ;
4084         *=KBDORG
4085 ;
4086 PWRONA: LDA     #$FF
4087         STA     CH
4088         LDA     MEMTOP+1
4089         AND     #$F0        ;INSURE 4K PAGE BOUNDARY
4090         STA     RAMTOP
4091         LDA     #$40        ;DEFAULT TO UPPER CASE ALPHA AT PWRON
4092         STA     SHFLOK
4093         RTS                 ;POWER ON COMPLETED
4094         .PAGE
4095 ;
4096 ;
4097 ; BEGIN DISPLAY HANDLER OPEN PROCESSING
4098 ;
4099 DOPEN:  LDA     ICAX2Z      ;GET AUX 2 BYTE
4100         AND     #$F
4101         BNE     OPNCOM      ;IF MODE ZERO, CLEAR ICAX1Z
4102 EOPEN:  LDA     ICAX1Z      ;CLEAR "CLR INHIBIT" AND "MXD MODE" BITS
4103         AND     #$F
4104         STA     ICAX1Z
4105         LDA     #0
4106 OPNCOM: STA     DINDEX
4107         LDA     #$E0        ;INITIALIZE GLOBAL VBLANK RAM
4108         STA     CHBAS
4109         LDA     #2
4110         STA     CHACT
4111         STA     SDMCTL      ;TURN OFF DMA NEXT VBLANK
4112         LDA     #SUCCES
4113         STA     DSTAT       ;CLEAR STATUS
4114         LDA     #$C0        ;DO IRQEN
4115         ORA     POKMSK
4116         STA     POKMSK
4117         STA     IRQEN
4118         LDA     #0
4119         STA     TINDEX      ;TEXT INDEX MUST ALWAYS BE 0
4120         STA     ADRESS
4121         STA     SWPFLG
4122         STA     CRSINH      ;TURN CURSOR ON ATOPEN
4123         LDY     #14         ;CLEAR TAB STOPS
4124         LDA     #1          ;INIT TAB STOPS TO EVERY 8 CHARACTERS
4125 CLRTBS: STA     TABMAP,Y
4126         DEY
```

```
4127          BPL     CLRTBS
4128          LDX     #4           ;LOAD COLOR REGISTERS
4129 DOPEN8: LDA     COLRTB,X
4130          STA     COLOR0,X
4131          DEX
4132          BPL     DOPEN8
4133          LDY     RAMTOP       ;DO TXTMSC=$2C40 (IF MEMTOP=3000)
4134          DEY
4135          STY     TXTMSC+1
4136          LDA     #$60
4137          STA     TXTMSC
4138          LDX     DINDEX
4139          LDA     ANCONV,X     ;CONVERT IT TO ANTIC CODE
4140          BNE     DOPENA       ;IF ZERO, IT IS ILLEGAL
4141 OPNERR: LDA     #BADMOD      ;SET ERROR STATUS
4142          STA     DSTAT
4143 DOPENA: STA     HOLD1
4144          LDA     RAMTOP       ;SET UP AN INDIRECT POINTER
4145          STA     ADRESS+1
4146          LDY     ALOCAT,X     ;ALLOCATE N BLOCKS OF 40 BYTES
4147 DOPEN1: LDA     #40
4148          JSR     DBSUB
4149          DEY
4150          BNE     DOPEN1
4151          LDA     GPRIOR       ;ICLEAR GTIA MODES
4152          AND     #$3F
4153          STA     OPNTMP+1
4154          TAY
4155          CPX     #8           ;TEST IF 320X1
4156          BCC     NOT8
4157          TXA                  ;GET 2 LOW BITS
4158          ROR     A
4159          ROR     A
4160          ROR     A
4161          AND     #$C0         ;NOW 2 TOP BITS
4162          ORA     OPNTMP+1
4163          TAY
4164          LDA     #16          ;SUBTRACT 16 MORE FOR PAGE BOUNDARY
4165          JSR     DBSUB
4166          CPX     #11          ;TEST MODE 11
4167          BNE     NOT8         ;IF MODE = 11
4168          LDA     #6           ;PUT GTIA LUM VALUE INTO BACKGROUND REGISTER
4169          STA     COLOR4
4170 NOT8:   STY     GPRIOR       ;STORE NEW PRIORITY
4171          LDA     ADRESS       ;SAVE MEMORY SCAN COUNTER ADDRESS
4172          STA     SAVMSC
4173          LDA     ADRESS+1
4174          STA     SAVMSC+1
4175 VBWAIT: LDA     VCOUNT       ;WAIT FOR NEXT VBLANK BEFORE MESSING
4176          CMP     #$7A         ;WITH THE DISPLAY LIST
4177          BNE     VBWAIT
4178          JSR     DBDEC        ;START PUTTING DISPLAY LIST RIGHT UNDER RAM
4179          LDA     PAGETB,X     ;TEST IF DISPLAY LIST WILL BE IN TROUBLE
4180          BEQ     NOMOD        ;OF CROSSING A 256 BYTE PAGE BOUNDARY
4181          LDA     #$FF         ;IF SO, DROP DOWN A PAGE
4182          STA     ADRESS
4183          DEC     ADRESS+1
4184 NOMOD:  LDA     ADRESS       ;SAVE END OF DISPLAY LIST FOR LATER
4185          STA     SAVADR
```

```
4186         LDA     ADRESS+1
4187         STA     SAVADR+1
4188         JSR     DBDDEC      ;(DOUBLE BYTE DOUBLE DECREMENT)
4189         LDA     #$41        ;(ANTIC) WAIT FOR VBLANK AND JMP TO TOP
4190         JSR     STORE
4191         STX     OPNTMP
4192         LDA     #24         ;INITIALIZE BOTSCR
4193         STA     BOTSCR
4194         LDA     DINDEX      ;DISALLOW MIXED MODE IF MODE.GE.9
4195         CMP     #9
4196         BCS     NOTMXD
4197         LDA     ICAX1Z      ;TEST MIXED MODE
4198         AND     #$10
4199         BEQ     NOTMXD
4200         LDA     #4
4201         STA     BOTSCR
4202         LDX     #2          ;ADD 4 LINES OF TEXT AT BOTTOM OF SCREEN
4203 DOPEN2: LDA     #2
4204         JSR     STORE
4205         DEX
4206         BPL     DOPEN2
4207         LDY     RAMTOP      ;RELOAD MSC FOR TEXT
4208         DEY
4209         TYA
4210         JSR     STORE
4211         LDA     #$60
4212         JSR     STORE
4213         LDA     #$42
4214         JSR     STORE
4215         CLC
4216         LDA     #MXDMDE-NUMDLE ;POINT X AT MIXED MODE TABLE
4217         ADC     OPNTMP
4218         STA     OPNTMP
4219 NOTMXD: LDY     OPNTMP
4220         LDX     NUMDLE,Y    ;GET NUMBER OF DISPLAY LIST ENTRIES
4221 DOPEN3: LDA     HOLD1       ;STORE N DLE'S
4222         JSR     STORE
4223         DEX
4224         BNE     DOPEN3
4225         LDA     DINDEX      ;DO THE MESSY 320X1 PROBLEM
4226         CMP     #8
4227         BCC     DOPEN5
4228         LDX     #93         ;GET REMAINING NUMBER OF DLE'S
4229         LDA     RAMTOP      ;RELOAD MEMORY SCAN COUNTER
4230         SEC
4231         SBC     #$10
4232         JSR     STORE
4233         LDA     #0
4234         JSR     STORE
4235         LDA     #$4F        ;(ANTIC) RELOAD MSC CODE
4236         JSR     STORE
4237 DOPEN4: LDA     HOLD1       ;DO REMAINING DLE'S
4238         JSR     STORE
4239         DEX
4240         BNE     DOPEN4
4241 DOPEN5: LDA     SAVMSC+1    ;POLISH OFF DISPLAY LIST
4242         JSR     STORE
4243         LDA     SAVMSC
4244         JSR     STORE
```

```
4245         LDA     HOLD1
4246         ORA     #$40
4247         JSR     STORE
4248         LDA     #$70         ;24 BLANK LINES
4249         JSR     STORE
4250         LDA     #$70
4251         JSR     STORE
4252         LDA     ADRESS       ;SAVE DISPLAY LIST ADDRESS
4253         STA     SDLSTL
4254         LDA     ADRESS+1
4255         STA     SDLSTL+1
4256         LDA     #$70         ;ADD LAST BLANK LINE ENTRY
4257         JSR     STORE        ;POSITION ADRESS=SDLSTL-1
4258         LDA     ADRESS       ;STORE NEW MEMTOP
4259         STA     MEMTOP
4260         LDA     ADRESS+1
4261         STA     MEMTOP+1
4262         LDA     SAVADR
4263         STA     ADRESS
4264         LDA     SAVADR+1
4265         STA     ADRESS+1
4266         LDA     SDLSTL+1
4267         JSR     STORE
4268         LDA     SDLSTL
4269         JSR     STORE
4270         LDA     DSTAT        ;IF ERROR OCURRED ON ALLOCATION, OPEN THE ED
4271         BPL     DOPEN9
4272         PHA                  ;SAVE STATUS
4273         JSR     EOPEN        ;OPEN THE EDITOR
4274         PLA                  ;RESTORE STATUS
4275         TAY                  ;AND RETURN IT TO CIO
4276         RTS
4277 DOPEN9: LDA     ICAX1Z       ;TEST CLEAR INHIBIT BIT
4278         AND     #$20
4279         BNE     DOPEN7
4280         JSR     CLRSCR       ;CLEAR SCREEN
4281         STA     TXTROW       ;AND HOME TEXT CURSOR (AC IS ZERO)
4282         LDA     LMARGN
4283         STA     TXTCOL
4284 DOPEN7: LDA     #$22         ;EVERYTHING ELSE IS SET UP
4285         ORA     SDMCTL       ;SO TURN ON DMACTL
4286         STA     SDMCTL
4287         JMP     RETUR2
4288 ;
4289 ;
4290 GETCH:  JSR     RANGE        ;GETCH DOES INCRSR. GETPLT DOESN'T
4291         JSR     GETPLT
4292         JSR     INATAC       ;CONVERT INTERNAL CODE TO ATASCII
4293         JSR     INCRSB
4294         JMP     RETUR1
4295 GETPLT: JSR     CONVRT       ;CONVERT ROW/COLUMN TO ADRESS
4296         LDA     (ADRESS),Y
4297         AND     DMASK
4298 SHIFTD: LSR     SHFAMT       ;SHIFT DATA DOWN TO LOW BITS
4299         BCS     SHIFT1
4300         LSR     A
4301         BPL     SHIFTD       ;(UNCONDITIONAL)
4302 SHIFT1: STA     CHAR
4303         CMP     #0           ;RESTORE FLAGS ALSO
```

```
4304         RTS
4305 ;
4306 ;
4307 ;
4308 OUTCH:  STA     ATACHR
4309         JSR     RANGE
4310 ;       JSR     OFFCRS
4311 OUTCHA: LDA     ATACHR      ;TEST FOR CLEAR SCREEN
4312         CMP     #CLRCOD
4313         BNE     OUTCHE
4314         JSR     CLRSCR
4315         JMP     RETUR2
4316 OUTCHE: LDA     ATACHR      ;TEST FOR CARRIAGE RETURN
4317         CMP     #CR
4318         BNE     OUTCHB
4319         JSR     DOCRWS      ;DO CR
4320         JMP     RETUR2
4321 OUTCHB: JSR     OUTPLT
4322         JSR     INCRSR
4323         JMP     RETUR2
4324 ;
4325 ;
4326 OUTPLT: LDA     SSFLAG      ;*****LOOP HERE IF START/STOP FLAG ISNON-0
4327         BNE     OUTPLT
4328         LDX     #2
4329 CRLOOP: LDA     ROWCRS,X    ;SAVE CURSOR LOCATION FOR DRAW LINE TO DRAW
4330         STA     OLDROW,X
4331         DEX
4332         BPL     CRLOOP
4333         LDA     ATACHR      ;CONVERT ATASCII(ATACHR) TO INTERNAL(CHAR)
4334         TAY                 ;SAVE ATACHR
4335         ROL     A
4336         ROL     A
4337         ROL     A
4338         ROL     A
4339         AND     #3
4340         TAX                 ;X HAS INDEX INTO ATAINT
4341         TYA                 ;RESTORE ATACHR
4342         AND     #$9F        ;STRIP OFF COLUMN ADDRESS
4343         ORA     ATAINT,X    ;OR IN NEW COLUMN ADDRESS
4344 OUTCH2: STA     CHAR
4345         JSR     CONVRT
4346         LDA     CHAR
4347 SHIFTU: LSR     SHFAMT      ;SHIFT UP TO PROPER POSITION
4348         BCS     SHIFT2
4349         ASL     A
4350         JMP     SHIFTU
4351 SHIFT2: AND     DMASK
4352         STA     TMPCHR      ;SAVE SHIFTED DATA
4353         LDA     DMASK       ;INVERT MASK
4354         EOR     #$FF
4355         AND     (ADRESS),Y  ;MASK OFF OLD DATA
4356         ORA     TMPCHR      ;OR IN NEW DATA
4357         STA     (ADRESS),Y
4358         RTS
4359 ;
4360 ;
4361 RETUR2: JSR     GETPLT      ;DO CURSOR ON THE WAY OUT
4362         STA     OLDCHR
```

```
4363         LDX     DINDEX      ;GRAPHICS HAVE INVISIBLE CURSOR
4364         BNE     RETUR1
4365         LDX     CRSINH      ;TEST CURSOR INHIBIT
4366         BNE     RETUR1
4367         EOR     #$80        ;TOGGLE MSB
4368         JSR     OUTCH2      ;DISPLAY IT
4369 RETUR1: LDY     DSTAT       ;RETURN TO CIO WITH STATUS IN Y
4370         LDA     #SUCCES
4371         STA     DSTAT       ;SET STATUS= SUCCESSFUL COMPLETION
4372         LDA     ATACHR      ;PUT ATACHR IN AC FOR RETURN TO CIO
4373 NOFUNC: RTS                 ;(NON-EXISTENT FUNCTION RETURN POINT)
4374 ;
4375 ;
4376 ;
4377 ; END OF DISPLAY HANDLER
4378 ;
4379         .PAGE
4380 ;
4381 ;
4382 ;
4383 ;
4384 EGETCH: JSR     SWAP
4385         JSR     ERANGE
4386         LDA     BUFCNT      ;ANYTHING IN THE BUFFER?
4387         BNE     EGETC3      ;YES
4388         LDA     ROWCRS      ;NO, SO SAVE BUFFER START ADDRESS
4389         STA     BUFSTR
4390         LDA     COLCRS
4391         STA     BUFSTR+1
4392 EGETC1: JSR     KGETCH      ;LET'S FILL OUR BUFFER
4393         STY     DSTAT       ;SAVE KEYBOARD STATUS
4394         LDA     ATACHR      ;TEST FOR CR
4395         CMP     #CR
4396         BEQ     EGETC2
4397         JSR     DOSS        ;NO, GO PRINT IT
4398         JSR     SWAP        ;JSR DOSS DID SWAP SO SWAP BACK
4399         LDA     LOGCOL      ;BEEP IF NEARING LOGICAL COL 120
4400         CMP     #113
4401         BNE     EGETC6
4402         JSR     BELL
4403 EGETC6: JMP     EGETC1
4404 EGETC2: JSR     OFFCRS      ;GET BUFFER COUNT
4405         JSR     DOBUFC
4406         LDA     BUFSTR      ;RETURN A CHARACTER
4407         STA     ROWCRS
4408         LDA     BUFSTR+1
4409         STA     COLCRS
4410 EGETC3: LDA     BUFCNT
4411         BEQ     EGETC5
4412 EGETC7: DEC     BUFCNT      ;AND RETURN TILL BUFCNT=0
4413         BEQ     EGETC5
4414         LDA     DSTAT       ;IF ERR, LOOP ON EGETC7 UNTIL BUFR IS E1IPTIE
4415         BMI     EGETC7
4416         JSR     GETCH
4417         STA     ATACHR
4418         JMP     SWAP        ;AND RETURN WITHOUT TURNING CURSOR BACK ON
4419 EGETC5: JSR     DOCRWS      ;DO REAL CARRIAGE RETURN
4420         LDA     #CR         ;AND RETURN EOL
4421         STA     ATACHR
```

```
4422         JSR     RETUR2      ;TURN ON CURSOR THEN SWAP
4423         STY     DSTAT       ;SAVE KEYBOARD STATUS
4424         JMP     SWAP        ;AND RETURN THROUGH RETUR1
4425 ;
4426 JSRIND: JMP     (ADRESS)    ;J5R TO THIS CAUSES JSR INDIRECT
4427 ;
4428 EOUTCH: STA     ATACHR      ;SAVE ATASCII VALUE
4429         JSR     SWAP
4430         JSR     ERANGE
4431 DOSS:   JSR     OFFCRS      ;TURN OFF CURSOR
4432         JSR     TSTCTL      ;TEST FOR CONTROL CHARACTERS (Z=1 IF CTL)
4433         BEQ     EOUTC5
4434 EOUTC6: ASL     ESCFLG      ;ESCFLG ONLY WORKS ONCE
4435         JSR     OUTCHE
4436 ERETN:  JMP     SWAP        ;AND RETURN THROUGH RETUR1
4437 EOUTC5: LDA     DSPFLG      ;DO DSPFLG AND ESCFLC
4438         ORA     ESCFLG
4439         BNE     EOUTC6      ;IF NON-0 DISPLAY RATHER THAN EXECUTE IT
4440         ASL     ESCFLG
4441         INX                 ;PROCESS CONTROL CHARACTERS
4442         LDA     CNTRLS,X    ;GET DISPLACEMENT INTO ROUTINE
4443         STA     ADRESS
4444         LDA     CNTRLS+1,X  ;GET HIGH BYTE
4445         STA     ADRESS+1
4446         JSR     JSRIND      ;DO COMPUTED JSR
4447         JSR     RETUR2      ;DO CURSOR
4448         JMP     SWAP        ;ALL DONE SO RETURN THROUGH RETUR1
4449 ;
4450 ;
4451 ;
4452 ;
4453 ; END SCREEN EDITOR.
4454 ;
4455 ;
4456 ; BEGIN KEYBOARD HANDLER
4457 ;
4458 ;
4459 ;
4460 ;
4461 KGETC2: LDA     #$FF
4462         STA     CH
4463 KGETCH: LDA     ICAX1Z      ;TEST LSB OF AUX1 FOR SPECIAL EDITOR READ MO
4464         LSR     A
4465         BCS     GETOUT
4466         LDA     #BRKABT
4467         LDX     BRKKEY      ;TEST BREAK
4468         BEQ     K7          ;IF BREAK, PUT BRKABT IN DSTAT AND CRIN ATA
4469         LDA     CH
4470         CMP     #$FF
4471         BEQ     KGETCH
4472         STA     HOLDCH      ;SAVE CH FOR SHIFT LOCK PROC
4473         LDX     #$FF        ;"CLEAR" CH
4474         STX     CH
4475         JSR     CLICK       ;DO KEYBOARD AUDIO FEEDBACK (A IS OK)
4476 KGETC3: TAX                 ;DO ASCCON
4477         CPX     #$C0        ;TEST FOR CTL & SHIFT TOGETHER
4478         BCC     ASCC01
4479         LDX     #3          ;BAD CODE
4480 ASCC01: LDA     ATASCI,X
```

```
4481         STA     ATACHR      ;DONE
4482         CMP     #$80        ;DO NULLS
4483         BEQ     KGETC2
4484         CMP     #$81        ;CHECK ATARI KEY
4485         BNE     KGETC1
4486         LDA     INVFLG
4487         EOR     #$80
4488         STA     INVFLG
4489         JMP     KGETC2      ;DONT RETURN A VALUE
4490 KGETC1: CMP     #$82        ;CAPS/LOWER.
4491         BNE     K1
4492         LDA     #0          ;CLEAR SHFLOK
4493         STA     SHFLOK
4494         BEQ     KGETC2
4495 K1:     CMP     #$83        ;SHIFT CAPS/LOWER
4496         BNE     K2
4497         LDA     #$40
4498         STA     SHFLOK      ;SHIFT BIT
4499         BNE     KGETC2
4500 K2:     CMP     #$84        ;CNTL CAPS/LOWER
4501         BNE     K3
4502         LDA     #$80        ;CNTL BIT
4503         STA     SHFLOK
4504         BNE     KGETC2
4505 K3:     CMP     #$85        ;DO EOF
4506         BNE     K6
4507         LDA     #EOFERR
4508 K7:     STA     DSTAT
4509         STA     BRKKEY      ;RESTORE BREAK
4510 GETOUT: LDA     #CR         ;PUT CR IN ATACHR
4511         BNE     K8          ;(UNCONDITIONAL)
4512 K6:     LDA     HOLDCH      ;PROCESS SHIFT LOCKS
4513         CMP     #$40        ;REGULAR SHIFT AND CONTROL TAKE PRECEDENCE
4514         BCS     K5          ;OVER LOCK
4515         LDA     ATACHR      ;TEST FOR ALPHA
4516         CMP     #$61        ;LOWER CASE A
4517         BCC     K5          ;NOT ALPHA IF LT
4518         CMP     #$7B        ;LOWER CASE Z+1
4519         BCS     K5          ;NOT ALPHA IF GE
4520         LDA     SHFLOK      ;DO SHIFT/CONTROL LOCK
4521         BEQ     K5          ;IF NO LOCK. DONT RE-DO IT
4522         ORA     HOLDCH
4523         JMP     KGETC3      ;DO RETRY
4524 K5:     JSR     TSTCTL      ;DONT INVERT M58 OF CONTROL CHARACTERS
4525         BEQ     K4
4526         LDA     ATACHR
4527         EOR     INVFLG
4528 K8:     STA     ATACHR
4529 K4:     JMP     RETUR1      ;ALL DONE
4530 ;
4531 ;
4532         .PAGE
4533 ;
4534 ;
4535 ;CONTROL CHARACTER PROCESSORS
4536 ;
4537 ESCAPE: LDA     #$80        ;SET ESCAPE FLAG
4538         STA     ESCFLG
4539         RTS
```

```
4540 CRSRUP: DEC     ROWCRS
4541         BPL     COMRET
4542         LDX     BOTSCR      ;WRAPAROUND
4543         DEX
4544 UPDNCM: STX     ROWCRS
4545 COMRET: JMP     STRBEG      ;CULVERT ROW AND COL TO LOGCOL AND RETURN
4546 CRSRDN: INC     ROWCRS
4547         LDA     ROWCRS
4548         CMP     BOTSCR
4549         BCC     COMRET
4550         LDX     #0
4551         BEQ     UPDNCM      ;(UNCONDITIONAL)
4552 CRSRLF: DEC     COLCRS
4553         LDA     COLCRS
4554         BMI     CRSRL1      ;(IF LMARGN=0, THIS ELIMINATES PROBLEM CASE)
4555         CMP     LMARGN
4556         BCS     COMRE1
4557 CRSRL1: LDA     RMARGN
4558 LFRTCM: STA     COLCRS
4559 COMRE1: JMP     DOLCOL      ;COLVERT ROW AND COL TO LOGCOL AND RETURN
4560 CRSRRT: INC     COLCRS
4561         LDA     COLCRS
4562         CMP     RMARGN
4563         BCC     COMRE1
4564         BEQ     COMRE1      ;(CAUSE OLE)
4565         LDA     LMARGN
4566         JMP     LFRTCM      ;UNCONDITIONAL TO COMMON STORE
4567 CLRSCR: JSR     PUTMSC
4568         LDY     #0
4569         TYA                 ;PUT 0 IN THEAC
4570 CLRSC2: STA     (ADRESS),Y  ;(AC IS ZERO)
4571         INY
4572         BNE     CLRSC2
4573         INC     ADRESS+1
4574         LDX     ADRESS+1
4575         CPX     RAMTOP
4576         BCC     CLRSC2
4577         LDA     #$FF        ;CLEAN UP LOGICAL LINE BITMAP
4578 CLRSC3: STA     LOGMAP,Y    ;(Y IS ZERO AFTER CLRSC2 LOOP)
4579         INY
4580         CPY     #4
4581         BCC     CLRSC3
4582 HOME:   JSR     COLCR       ;PLACE COLCRS AT LEFT EDGE
4583         STA     LOGCOL
4584         STA     BUFSTR+1
4585         LDA     #0
4586         STA     ROWCRS
4587         STA     COLCRS+1
4588         STA     BUFSTR
4589         RTS
4590 ;
4591 BS:     LDA     LOGCOL      ;BACKSPACE
4592         CMP     LMARGN
4593         BEQ     BS1
4594 BSA:    LDA     COLCRS      ;LEFT EDGE?
4595         CMP     LMARGN
4596         BNE     BS3         ;NO
4597         JSR     DELTIM      ;YES, SEE IF LINE SHOULD BE DELETED
4598 BS3:    JSR     CRSRLF
```

```
4599         LDA     COLCRS
4600         CMP     RMARGN
4601         BNE     BS2
4602         LDA     ROWCRS
4603         BEQ     BS2
4604         JSR     CRSRUP
4605 BS2:    LDA     #$20         ;MAKE BACKSPACE DESTRUCTIVE
4606         STA     ATACHR
4607         JSR     OUTPLT
4608 BS1:    JMP     DOLCOL       ;AND RETURN
4609 TAB:    JSR     CRSRRT       ;BEGIN SEARCH
4610         LDA     COLCRS       ;TEST FOR NEW LINE
4611         CMP     LMARGN
4612         BNE     TAB1         ;NO
4613         JSR     DOCR         ;DO CARRIAGE RETURN
4614         JSR     LOGGET       ;CHECK IF END OF LOGICAL LINE
4615         BCC     TAB1         ;NO, CONTINUE
4616         BCS     TAB2         ;(UNCONDITIONAL)
4617 TAB1:   LDA     LOGCOL       ;CHECK FOR TAB STOP
4618         JSR     BITGET
4619         BCC     TAB          ;NO, SO KEEP LOOKING
4620 TAB2:   JMP     DOLCOL       ;CULVERT ROW AND COL TO LOGCOL AND RETURN
4621 SETTAB: LDA     LOGCOL
4622         JMP     BITSET       ;SET BIT IN MAP AND RETURN
4623 CLRTAB: LDA     LOGCOL
4624         JMP     BITCLR       ;CLEAR " " " " "
4625 INSCHR: JSR     PHACRS
4626         JSR     GETPLT       ;GET CHARACTER UNDER CURSOR
4627         STA     INSDAT
4628         LDA     #0
4629         STA     SCRFLG
4630 INSCH4: JSR     OUTCH2       ;STORE DATA
4631         LDA     LOGCOL       ;SAVE LOGCOL: IF AFTER INCRSA LOGCOL IS
4632         PHA                  ;< THAN IT IS NOW, END LOOP
4633         JSR     INCRSA       ;SPECIAL INCRSR ENTRY POINT
4634         PLA
4635         CMP     LOGCOL
4636         BCS     INSCH3       ;QUIT
4637 INSCH1: LDA     INSDAT       ;KEEP GOING
4638         PHA
4639         JSR     GETPLT
4640         STA     INSDAT
4641         PLA
4642         JMP     INSCH4
4643 INSCH3: JSR     PLACRS
4644 INSCH6: DEC     SCRFLG
4645         BMI     INSCH5       ;IF SCROLL OCCURRED
4646         DEC     ROWCRS       ;MOVE CURSOR UP
4647         BNE     INSCH6       ;(UNCOND) CONTINUE UNTIL SCRFLG IS MINUS
4648 INSCH5: JMP     DOLCOL       ;CULVERT ROW AND COL TO LOGCOL AND RETURN
4649 ;
4650 ;
4651 DELCHR: JSR     PHACRS
4652 DELCH1: JSR     CONVRT       ;GET DATA TO THE RIGHT OF THE CURSOR
4653         LDA     ADRESS
4654         STA     SAVADR       ;SAVE ADRESS TO KNOW WHERE TO PUT DATA
4655         LDA     ADRESS+1
4656         STA     SAVADR+1
4657         LDA     LOGCOL
```

```
4658         PHA
4659         JSR     INCRSB      ;PUT CURSOR OVER NEXT CHARACTER
4660         PLA
4661         CMP     LOGCOL      ;TEST NEW LOGCOL AGAINST OLD LOGCOL
4662         BCS     DELCH2      ;IF OLD.GE.NEW THEN QUIT
4663         LDA     ROWCRS      ;IS ROW OFF SCREEN?
4664         CMP     BOTSCR
4665         BCS     DELCH2      ;YES, SO QUIT
4666         JSR     GETPLT      ;GET DATA UNDER CURSOR
4667         LDY     #0
4668         STA     (SAVADR),Y  ;PUT IT IN PREVIOUS POSITION
4669         BEQ     DELCH1      ;AND LOOP (UNCONDITIONAL)
4670 DELCH2: LDY     #0
4671         TYA
4672         STA     (SAVADR),Y  ;CLEAR THE LAST POSITION
4673         JSR     DELTIA      ;TRY TO DELETE A LINE
4674         JSR     PLACRS
4675         JMP     DOLCOL      ;AND RETURN
4676 INSLIN: SEC                 ;INSLIN PUTS "1" INTO BIT MAP
4677 INSLIA: JSR     EXTEND      ;ENTRY POINT FOR C=0
4678         LDA     LMARGN      ;DO CARRIAGE RETURN (NO LF)
4679         STA     COLCRS
4680         JSR     CONVRT      ;GET ADDRESS
4681         LDA     ADRESS      ;SET UP TO=40+FROM (FROM = CURSOR)
4682         STA     FRMADR
4683         CLC
4684         ADC     #40
4685         STA     TOADR
4686         LDA     ADRESS+1
4687         STA     FRMADR+1
4688         ADC     #0
4689         STA     TOADR+1
4690         LDX     ROWCRS      ;SET UP LOOP COUNTER
4691         CPX     #23
4692         BEQ     INSLI2
4693 INSLI1: JSR     MOVLIN
4694         INX
4695         CPX     #23
4696         BNE     INSLI1
4697 INSLI2: JSR     CLRLIN      ;CLEAR CURRENT LINE
4698         JMP     DOLCOL      ;COLVERT ROW AND COL TO LOGCOL AND RETURN
4699 DELLIN: JSR     DOLCOL      ;GET BEGINNING OF LOG LINE (HOLD1)
4700 DELLIA: LDY     HOLD1       ;SQUEEZE BIT MAP
4701         STY     ROWCRS      ;PUT CURSOR THERE
4702 DELLIB: LDY     ROWCRS
4703 DELLI1: TYA
4704         SEC
4705         JSR     LO2GET      ;GET NEXT BIT
4706         PHP
4707         TYA
4708         CLC
4709         ADC     #120
4710         PLP
4711         JSR     BITPUT      ;WRITE IT OVER PRESENT BIT
4712         INY
4713         CPY     #24
4714         BNE     DELLI1      ;LOOP
4715         LDA     LOGMAP+2    ;SET LSB
4716         ORA     #1
```

```
4717         STA     LOGMAP+2
4718 DELLI2: LDA     LMARGN      ;DELETE LINE OF DATA USING PART OF SCROLL
4719         STA     COLCRS      ;CR NO LF
4720         JSR     CONVRT
4721         JSR     SCROL1
4722         JSR     LOGGET      ;TEST NEXT LINE FOR CONTINUATION
4723 ; IS IT A NEW LOG LINE?
4724         BCC     DELLIB      ;NO SO DELETE ANOTHER
4725         JMP     DOLCOL      ;YES SO DOLCOL AND RETURN
4726 BELL:   LDY     #$20
4727 BELL1:  JSR     CLICK
4728         DEY
4729         BPL     BELL1
4730         RTS
4731         .PAGE
4732 ;
4733 ;
4734 ; ROUTINES
4735 ;
4736 ;
4737 ; DOUBLE BYTE DECREMENT OF INDIRECT POINTER
4738 ; INCLUDING DB SUBTRACT AND DB DOUBLE DECREMENT
4739 ;
4740 DBDDEC: LDA     #2
4741         BNE     DBSUB       ;(UNCONDITIONAL)
4742 ;
4743 ; STORE DATA INDIRECT AND DECREMENT POINTER
4744 ; (PLACED HERE TO SAVE JMP DBDEC AFTER STORE)
4745 STORE:  LDY     DSTAT       ;RETURN ON ERROR
4746         BMI     STROK
4747         LDY     #0
4748 STOREI: STA     (ADRESS),Y
4749 ;       JMP     DBDEC       ;DECREMENT AND RETURN
4750 ;
4751 DBDEC:  LDA     #1
4752 DBSUB:  STA     SUBTMP
4753         LDA     DSTAT       ;RETURN ON ERROR
4754         BMI     STROK
4755         LDA     ADRESS
4756         SEC
4757         SBC     SUBTMP
4758         STA     ADRESS
4759         BCS     DBSUB1
4760         DEC     ADRESS+1
4761 DBSUB1: LDA     APPMHI+1    ;MARE SURE NOTHING EVER OVERWRITES APPMHI
4762         CMP     ADRESS+1
4763         BCC     STROK       ;OK
4764         BNE     STRERR      ;ERROR
4765         LDA     APPMHI
4766         CMP     ADRESS
4767         BCC     STROK
4768 STRERR: LDA     #SCRMEM     ;SHOW MEM TOO SMALL FOR SCREEN ERROR
4769         STA     DSTAT
4770 STROK:  RTS
4771 ;
4772 ;
4773 ;
4774 ; CONVERT ROW/COLUMN CURSOR INTO REAL ADDRESS (FROM SAVMSC ON UP)
4775 ;
```

```
4776 CONVRT: LDA     ROWCRS       ;SAVE CURSOR
4777         PHA
4778         LDA     COLCRS
4779         PHA
4780         LDA     COLCRS+1
4781         PHA
4782         JSR     PUTMSC
4783         LDA     ROWCRS       ;PUT 10*ROWCRS INTO MLTTMP
4784         STA     MLTTMP
4785         LDA     #0
4786         STA     MLTTMP+1
4787         LDA     MLTTMP       ;QUICK X8
4788         ASL     A
4789         ROL     MLTTMP+1
4790         STA     HOLD1        ;(SAVE 2X VALUE)
4791         LDY     MLTTMP+1
4792         STY     HOLD2
4793         ASL     A
4794         ROL     MLTTMP+1
4795         ASL     A
4796         ROL     MLTTMP+1
4797         CLC                  ;ADD IN 2X
4798         ADC     HOLD1
4799         STA     MLTTMP
4800         LDA     MLTTMP+1
4801         ADC     HOLD2
4802         STA     MLTTMP+1
4803         LDX     DINDEX       ;NOW SHIFT MLTTMP LEFT DHLINE TIMES TO FINIS
4804         LDY     DHLINE,X     ;MULTIPLY
4805 CONVR1: DEY                  ;LOOP N TIMES
4806         BMI     CONVR2
4807         ASL     MLTTMP
4808         ROL     MLTTMP+1
4809         JMP     CONVR1
4810 CONVR2: LDY     DIV2TB,X     ;NOW DIVIDE HCRSR TO ACCOUNT FOR PARTIAL BYT
4811         LDA     COLCRS
4812         LDX     #7           ;* TRICKY *
4813 CONVR3: DEY
4814         BMI     CONVR4
4815         DEX
4816         LSR     COLCRS+1
4817         ROR     A
4818         ROR     TMPLBT       ;SAVE LOW BITS FOR MASK
4819         JMP     CONVR3
4820 CONVR4: INY                  ;SO Y IS ZERO UPON RETURN FROM THIS ROUTINE
4821         CLC
4822         ADC     MLTTMP       ;ADD SHIFTED COLCRS TO MLTThP
4823         STA     MLTTMP
4824         BCC     CONVR5
4825         INC     MLTTMP+1
4826 CONVR5: SEC                  ;* TRICKY *
4827 CONVR6: ROR     TMPLBT       ;SLIDE A "1" UP AGAINST LOW BITS (CONTINUE T
4828         CLC
4829         DEX                  ;AND FINISH SHIFT SO LOW BITS ARE
4830         BPL     CONVR6       ;RIGHT JUSTIFIED.
4831         LDX     TMPLBT       ;TMPLBT IS NOW THE INDEX INTO DMASKTB
4832         LDA     MLTTMP       ;PREPARE FOR RETURN
4833         CLC
4834         ADC     ADRESS
```

```
4835         STA     ADRESS
4836         STA     OLDADR      ;REMEMBER THIS ADDRESS FOR CURSOR
4837         LDA     MLTTMP+1
4838         ADC     ADRESS+1
4839         STA     ADRESS+1
4840         STA     OLDADR+1
4841         LDA     DMASKT,X
4842         STA     DMASK
4843         STA     SHFAMT
4844         PLA
4845         STA     COLCRS+1
4846         PLA
4847         STA     COLCRS
4848         PLA
4849         STA     ROWCRS
4850         RTS
4851 ;
4852 ;
4853 ; INCREMENT CURSOR AND DETECT BOTH END OF LINE AND END OF SCREEN
4854 ;
4855 INCRSB: LDA     #0          ;NON-EXTEND ENTRY POINT
4856         BEQ     INCREC
4857 INCRSR: LDA     #$9B        ;SPECIAL CASE ELIMINATOR
4858 INCREC: STA     INSDAT
4859 INCRSA: INC     LOGCOL      ;(INSCHR ENTRY POINT)
4860         INC     COLCRS
4861         BNE     INCRS2      ;DO HIGH BYTE
4862         INC     COLCRS+1
4863 INCRS2: LDA     COLCRS      ;TEST END OF LINE
4864         LDX     DINDEX
4865         CMP     COLUMN,X    ;TEST TABLED VALUE FOR ALL SCREEN MODES
4866         BEQ     INC2A       ;DO CR IF EQUAL
4867         CPX     #0          ;MODE 0?
4868         BNE     INCRS3      ;IF NOT. JUST RETURN
4869         CMP     RMARGN      ;TEST AGAINST RMARGN
4870         BEQ     INCRS3      ;EGUAL IS OK
4871         BCS     INC2A       ;IF GREATER THAN, DO CR
4872 INCRS3: RTS
4873 INC2A:  CPX     #8          ;CHECK MODE
4874         BCC     DOCR1       ;NOT 320X1 $O DO IT
4875         LDA     COLCRS+1    ;TEST MED
4876         BEQ     INCRS3      ;ONLY AT 64 SO DON'T DO IT
4877 DOCR1:  LDA     DINDEX      ;DON'T MESS WITH LOGMAP IF NO MODE ZERO
4878         BNE     DOCR
4879         LDA     LOGCOL      ;TEST LINE OVERRUN
4880         CMP     #81
4881         BCC     DOCR1B      ;IF LESS THAN 81 IT IS DEFINITELY NOT LINE 3
4882         LDA     INSDAT
4883         BEQ     DOCR        ;ONLY DO LOG LINE OVERFLOW IF INSDAT <>0
4884         JSR     DOCRWS      ;LOG LINE OVERFLOW IS SPECIAL CASE
4885         JMP     INCRS1      ;RETURN
4886 DOCR1B: JSR     DOCR        ;GET IT OVER WITH
4887         LDA     ROWCRS
4888         CLC                 ;TEST LOGICAL LINE BIT MAP
4889         ADC     #120
4890         JSR     BITGET
4891         BCC     DOCR1A      ;DON'T EXTEND IF OVERRUN IS INTO MIDDLE OF L
4892         LDA     INSDAT      ;DON'T EXTEND IF INSDAT IS ZERO
4893         BEQ     DOCR1A      ;(INSCHR SPECIAL CASE)
```

```
4894         CLC                    ;INSERT "0" INTO BIT MAP
4895         JSR     INSLIA
4896 DOCR1A: JMP     DOLCOL      ;CONVERT ROW AND COL TO LOGCOL AND RETURN
4897 NOSCRL: LDA     #0          ;DOCR WITHOUT SCROLL
4898         BEQ     NOSCR1      ;(UNCONDITIONAL)
4899 DOCRWS: LDA     #$9B        ;DOCR WITH SCROLLING (NORMAL MODE)
4900 NOSCR1: STA     INSDAT
4901 DOCR:   JSR     COLCR       ;PLACE COLCRS AT LEFT EDGE
4902         LDA     #0
4903         STA     COLCRS+1
4904         INC     ROWCRS
4905 DOCR2:  LDX     DINDEX
4906         LDY     #24         ;SET UP SCROLL LOOP COUNTER
4907         BIT     SWPFLG
4908         BPL     DOCR2A      ;BRANCH IF NORMAL
4909         LDY     #4
4910         TYA
4911         BNE     DOCR2B      ;(UNCONDITIONAL)
4912 DOCR2A: LDA     NOROWS,X    ;GET NO OF ROWS
4913 DOCR2B: CMP     ROWCRS
4914         BNE     INCRS1
4915         STY     HOLD3
4916         TXA                 ;DON'T SCROLL IF MODE <> 0
4917         BNE     INCRS1
4918         LDA     INSDAT      ;OR IF INSDAT = 0
4919         BEQ     INCRS1
4920 ;       LDA     INSDAT      IF INSDAT <> $9B THEN ROLL IN A 0
4921         CMP     #$9B        ;TO EXTEND BOTTOM LOGICAL LINE
4922         SEC
4923         BEQ     DOCR4B
4924         CLC
4925 DOCR4B: JSR     SCROLL      ;LOOP SACK TO HERE IF >1 SCROLLS
4926         INC     SCRFLG
4927         DEC     BUFSTR      ;ROWS MOVE UP SO BUFSTR SHOULD TOO
4928         DEC     HOLD3
4929         LDA     LOGMAP
4930         SEC                 ;FOR PARTIAL LINES ROLL IN A "1"
4931         BPL     DOCR4B      ;AGAIN IF PARTIAL LOGICAL LINE
4932         LDA     HOLD3       ;PLACE CURSOR AT NEW LINE NEAR THE BOTTOM
4933         STA     ROWCRS
4934 INCRS1: JMP     DOLCOL      ;COLVERT ROW AND COL TO LOGCDL AND RETURN
4935 ;
4936 ;
4937 ; SUBEND: SUBTRACT ENDPT FROM ROWAC OR COLAC. (X=0 OR 2)
4938 ;
4939 SUBEND: SEC
4940         LDA     ROWAC,X
4941         SBC     ENDPT
4942         STA     ROWAC,X
4943         LDA     ROWAC+1,X
4944         SBC     ENDPT+1
4945         STA     ROWAC+1,X
4946         RTS
4947 ;
4948 ;
4949 ; RANGE: DO CURSOR RANGE TEST. IF ERROR, POP STACK TWICE AND JMP RETURN
4950 ;         (ERANGE IS EDITOR ENTRY POINT AND TEST IF EDITOR IS OPEN.
4951 ;          IF IT ISNT IT OPENS THE EDITOR AND CONTINUES)
4952 ;
```

```
4953 ERANGE: LDA     BOTSCR      ; IF BOTSCR=4
4954         CMP     #4
4955         BEQ     RANGE       ;THEN IT IS IN MIXED NODE AND OK
4956         LDA     DINDEX      ;IF MODE = 0
4957         BEQ     RANGE       ;THEN IT IS INEDITOR MODE AND OK
4958         JSR     EOPEN       ;IF NOT, OPEN EDITOR
4959 RANGE:  LDA     #39         ;***** RANGE CHECK RMARGN ***** SET UP AC
4960         CMP     RMARGN      ;***** RANGE CHECK RMARGN ***** COMPARE
4961         BCS     RANGE3      ;***** RANGE CHECK RMARGN ***** BRANCH GE
4962         STA     RMARGN      ;***** RANGE CHECK RMARGN ***** BAD SO STORE
4963 RANGE3: LDX     DINDEX
4964         LDA     NOROWS,X    ;CHECK ROWS
4965         CMP     ROWCRS
4966         BCC     RNGERR      ;(ERROR IF TABLE.GE.ROWCRS)
4967         BEQ     RNGERR
4968         CPX     #8          ;CHECK FOR 320X1
4969         BNE     RANGE1      ;SPECIAL CASE IT
4970         LDA     COLCRS+1
4971         BEQ     RNGOK       ;IF HIGH BYTE IS 0, COL IS OK
4972         CMP     #1
4973         BNE     RNGERR      ;IF >1, BAD
4974         BEQ     RANGE2      ;IF 1, GO CHECK LOWBYTE
4975 RANGE1: LDA     COLCRS+1    ;FOR OTHERS, NON-ZERO HIGH BYTE IS BAD
4976         BNE     RNGERR
4977 RANGE2: LDA     COLUMN,X    ;CHECK LOWBYTE
4978         CMP     COLCRS
4979         BCC     RNGERR
4980         BEQ     RNGERR
4981 RNGOK:  LDA     #SUCCES     ;SET STATUS OK
4982         STA     DSTAT
4983         LDA     #BRKABT     ;PREPARE BREAK ABORT STATUS
4984         LDX     BRKKEY      ;CHECK BREAK KEY FLAG
4985         STA     BRKKEY      ;'CLEAR' BREAK
4986         BEQ     RNGER2      ;IF BREAK, QUIT IMMEDIATELY AND RETURN TO CI
4987         RTS
4988 RNGERR: JSR     HOME        ;ON RANGE ERROR, BRING CURSOR BACK
4989         LDA     #CRSROR     ;SHOW CURSOR OVERRANGE ERROR
4990 RNGER2: STA     DSTAT
4991 RNGER1: PLA                 ;RESTORE STACK (THIS ROUTINE IS ALWAYS 1 LEV
4992         PLA                 ;AWAY FROM RETURN TO CIO)
4993         LDA     SWPFLG      ;IF SWAPPED. SWAP BACK
4994         BPL     RETUR3
4995         JSR     SWAPA       ;AND DONT DO RETUR1
4996 RETUR3: JMP     RETUR1      ;RETURN TO CIO
4997 ;
4998 ;
4999 ;
5000 ; OFFCRS: RESTORE OLD DATA UNDER CURSOR SO IT CAN BE MOVED
5001 ;
5002 OFFCRS: LDY     #0
5003         LDA     OLDCHR
5004         STA     (OLDADR),Y
5005         RTS
5006 ;
5007 ;
5008 ;
5009 ; BITMAP ROUTINES:
5010 ;
5011 ; BITCON: PUT MASK IN BITMSK AND INDEX IN X
```

```
5012 ; BITPUT: PUT CARRY INTO BITMAP
5013 ; BITROL: ROL CARRY INTO BOTTOM OF BiTMAP (SCROLL)
5014 ; BITSET: SET PROPER BIT
5015 ; BITCLR: CLEAR PROPER BIT
5016 ; BITGET: RETURN CARRY SET IF BIT IS THERE
5017 ; LOGGET: DO BITGET FOR LOGMAP INSTEAD OF TABMAP
5018 ;
5019 BITCON: PHA
5020         AND     #7
5021         TAX                     ;GET MASK
5022         LDA     MASKTB,X
5023         STA     BITMSK
5024         PLA                     ;PROCESS INDEX
5025         LSR     A
5026         LSR     A
5027         LSR     A
5028         TAX
5029         RTS
5030 ;
5031 ;
5032 BITROL: ROL     LOGMAP+2
5033         ROL     LOGMAP+1
5034         ROL     LOGMAP
5035         RTS
5036 ;
5037 ;
5038 BITPUT: BCC     BITCLR      ;AND RETURN
5039 ; OTHERWISE FALL THROUGH TO BITSET AND RETURN
5040 BITSET: JSR     BITCON
5041         LDA     TABMAP,X
5042         ORA     BITMSK
5043         STA     TABMAP,X
5044         RTS
5045 ;
5046 BITCLR: JSR     BITCON
5047         LDA     BITMSK
5048         EOR     #$FF
5049         AND     TABMAP,X
5050         STA     TABMAP,X
5051         RTS
5052 ;
5053 LOGGET: LDA     ROWCRS
5054 LO1GET: CLC
5055 LO2GET: ADC     #120
5056 BITGET: JSR     BITCON
5057         CLC
5058         LDA     TABMAP,X
5059         AND     BITMSK
5060         BEQ     BITGE1
5061         SEC
5062 BITGE1: RTS
5063 ;
5064 ;
5065 ;
5066 ;
5067 ; INATAC: INTERNAL(CHAR) TO ATASCII(ATACHR) CONVERSION
5068 ;
5069 INATAC: LDA     CHAR
5070         LDY     DINDEX      ;IF GRAPHICS MODES
```

```
5071         CPY     #3
5072         BCS     INATA1      ;THEN DON'T CHANGE CHAR
5073         ROL     A
5074         ROL     A
5075         ROL     A
5076         ROL     A
5077         AND     #3
5078         TAX
5079         LDA     CHAR
5080         AND     #$9F
5081         ORA     INTATA,X
5082 INATA1: STA     ATACHR
5083         RTS
5084 ;
5085 ;
5086 ;
5087 ; MOVLLN: MOVE 40 BYTES AT FRMADR TO TOADR SAVING OLD TOAOR
5088 ;       DATA IN THE LINBUF. THEN MAKE NEXT FRMADR
5089 ;       BE AT LINBUF FOR NEXT TRANSFER & TOADR=TOADR+40
5090 ;
5091 MOVLIN: LDA     #LINBUF/256 ;SET UP ADRESS=LINBUF$=247
5092         STA     ADRESS+1
5093         LDA     #LINBUF.AND.$FF
5094         STA     ADRESS
5095         LDY     #39
5096 MOVLI1: LDA     (TOADR),Y   ;SAVE TO DATA
5097         STA     TMPCHR
5098         LDA     (FRMADR),Y  ;STORE DATA
5099         STA     (TOADR),Y
5100         LDA     TMPCHR
5101         STA     (ADRESS),Y
5102         DEY
5103         BPL     MOVLI1
5104         LDA     ADRESS+1    ;SET UP FRMADR=LAST LINE
5105         STA     FRMADR+1
5106         LDA     ADRESS
5107         STA     FRMADR
5108         CLC                 ;ADD 40 TO TOADR
5109         LDA     TOADR
5110         ADC     #40
5111         STA     TOADR
5112         BCC     MOVLI2
5113         INC     TOADR+1
5114 MOVLI2: RTS
5115 ;
5116 ;
5117 ;
5118 ; EXTEND: EXTEND BIT MAP FROM ROWCRS (EXTEND LOGICAL LINE
5119 ;
5120 EXTEND: PHP                ;SAVE CARRY
5121         LDY     #23
5122 EXTEN1: TYA
5123         JSR     LO1GET
5124         PHP
5125         TYA
5126         CLC
5127         ADC     #121
5128         PLP
5129         JSR     BITPUT
```

```
5130 EXTEN3: DEY
5131         BMI     EXTEN4
5132         CPY     ROWCRS
5133         BCS     EXTEN1
5134 EXTEN4: LDA     ROWCRS
5135         CLC
5136         ADC     #120
5137         PLP
5138         JMP     BITPUT     ;STORE NEW LINE'S BIT AND RETURN
5139 ;
5140 ;
5141 ;
5142 ; CLRLIN: CLEAR LINE CURSOR IS ON
5143 ;
5144 CLRLIN: LDA     LMARGN
5145         STA     COLCRS
5146         JSR     CONVRT
5147         LDY     #39
5148         LDA     #0
5149 CLRLI1: STA     (ADRESS),Y
5150         DEY
5151         BPL     CLRLI1
5152         RTS
5153 ;
5154 ;
5155 ;
5156 ;
5157 ; SCROLL: SCROLL SCREEN
5158 ;
5159 SCROLL: JSR     BITROL      ;ROLL IN CARRY
5160         LDA     SAVMSC      ;SET UP WORKING REGISTERS
5161         STA     ADRESS
5162         LDA     SAVMSC+1
5163         STA     ADRESS+1
5164 SCROL1: LDY     #40         ;LOOP
5165         LDA     (ADRESS),Y
5166         LDX     RAMTOP      ;TEST FOR LAST LINE
5167         DEX
5168         CPX     ADRESS+1
5169         BNE     SCROL2
5170         LDX     #$D7
5171         CPX     ADRESS
5172         BCS     SCROL2
5173         LDA     #0          ;YES SO STORE ZERO DATA FOR THIS ENTIRE LINE
5174 SCROL2: LDY     #0
5175         STA     (ADRESS),Y
5176         INC     ADRESS
5177         BNE     SCROL1
5178         INC     ADRESS+1
5179         LDA     ADRESS+1
5180         CMP     RAMTOP
5181         BNE     SCROL1
5182         JMP     DOLCOL      ;AND RETURN
5183 ;
5184 ;
5185 ; DOLCOL: DO LOGICAL COLUMN FROM BITMAP AND COLCRS
5186 ;
5187 DOLCOL: LDA     #0          ;START WITH ZERO
5188         STA     LOGCOL
```

```
5189         LDA     ROWCRS
5190         STA     HOLD1
5191 DOLCO1: LDA     HOLD1        ;ADD IN ROW COMPONENT
5192         JSR     LO1GET
5193         BCS     DOLCO2       ;FOUND BEGINNING OF LINE
5194         LDA     LOGCOL       ;ADD 40 AND LOOK BAC ONE
5195         CLC
5196         ADC     #40
5197         STA     LOGCOL
5198         DEC     HOLD1        ;UP ONE LINE
5199         JMP     DOLCO1
5200 DOLCO2: CLC                  ;ADD IN COLCRS
5201         LDA     LOGCOL
5202         ADC     COLCRS
5203         STA     LOGCOL
5204         RTS
5205 ;
5206 ;
5207 ;
5208 ; DOBUFC: COMPUTE BUFFER COUNT AS THE NUMBER OF BYTES FROM
5209 ;         BUFSTR TO END OF LOGICAL LINE WITH TRAILING SPACES REMOVED
5210 ;
5211 DOBUFC: JSR     PHACRS
5212         LDA     LOGCOL
5213         PHA
5214         LDA     BUFSTR       ;START
5215         STA     ROWCRS
5216         LDA     BUFSTR+1
5217         STA     COLCRS
5218         LDA     #1
5219         STA     BUFCNT
5220 DOBUF1: LDX     #23          ;NORMAL
5221         LDA     SWPFLG       ;IF SWAPPED, ROW 3 IS THE LAST LINE ON SCREE
5222         BPL     DOB1
5223         LDX     #3
5224 DOB1:   CPX     ROWCRS       ;TEST IF CRSR IS AT LAST SCREEN POSITION
5225         BNE     DOBU1A
5226         LDA     COLCRS
5227         CMP     RMARGN
5228         BNE     DOBU1A
5229         INC     BUFCNT       ;YES, SO FAKE INCRSP TO AVOID SCROLLING
5230         JMP     D0BUF2
5231 DOBU1A: JSR     INCRSB
5232         INC     BUFCNT
5233         LDA     LOGCOL
5234         CMP     LMARGN
5235         BNE     DOBUF1       ;NOT YET EOL
5236         DEC     ROWCRS       ;BACK UP ONE INCRSR
5237         JSR     CRSRLF
5238 DOBUF2: JSR     GETPLT       ;TEST CURRENT COLUMN FOR NON-ZERO DATA
5239         BNE     DOBUF4       ;QUIT IF NON-ZERO
5240         DEC     BUFCNT       ;DECREMENT COUNTER
5241         LDA     LOGCOL       ;BEGINNING OF LOGICAL LINE YET?
5242         CMP     LMARGN
5243         BEQ     DOBUF4       ;YES, SO QUIT
5244         JSR     CRSRLF       ;BACK UP CURSOR
5245         LDA     COLCRS       ;IF LOGCOL=RMARGN, GO UP 1 ROW
5246         CMP     RMARGN
5247         BNE     DOBUF3
```

```
5248         DEC     ROWCRS
5249 DOBUF3: LDA     BUFCNT
5250         BNE     DOBUF2      ;LOOP UNLESS BUFCNT JUST WENT TO ZERO
5251 DOBUF4: PLA
5252         STA     LOGCOL
5253         JSR     PLACRS
5254         RTS
5255 ;
5256 ;
5257 ;
5258 ;
5259 ; STRBEG: MOVE BUFSTR TO BEGINNING OF LOGICAL LINE.
5260 ;
5261 STRBEG: JSR     DOLCOL      ;USE DOLCOL TO POINT HOLD1 AT BOL
5262         LDA     HOLD1
5263         STA     BUFSTR
5264         LDA     LMARGN
5265         STA     BUFSTR+1
5266         RTS
5267 ;
5268 ;
5269 ;
5270 ;
5271 ;
5272 ; DELTIM:  TIME TO DELETE A LINE IF IT IS EMPTY AND AN EXTENSION
5273 ;
5274 DELTIA: LDA     LOGCOL      ;IF LOGCOL<>LMARGN
5275         CMP     LMARGN      ;THEN DONT MOVE UP ONE
5276         BNE     DELTIG      ;LINE BEFORE TESTING DELTIM
5277         DEC     ROWCRS
5278 DELTIG: JSR     DOLCOL
5279 DELTIM: LDA     LOGCOL      ;TEST FOR EXTENSION
5280         CMP     LMARGN
5281         BEQ     DELTI3      ;NO
5282         JSR     CONVRT
5283         LDA     RMARGN      ;SET UP COUNT
5284         SEC
5285         SBC     LMARGN
5286         TAY
5287 DELTI1: LDA     (ADRESS),Y
5288         BNE     DELTI3      ;FOUND A NON-0 SD QUIT AND RETURN
5289         DEY
5290         BPL     DELTI1
5291 DELTI2: JMP     DELLIB      ;DELETE A LINE AND RETURN
5292 DELTI3: RTS
5293 ;
5294 ;
5295 ;
5296 ; TSTCTL: SEARCH CNTRLS TABLE TO SEE IF ATACHR IS A CNTL CHAR
5297 ;
5298 TSTCTL: LDX     #45         ;PREPARE TO SEARCH TABLE
5299 TSTCT1: LDA     CNTRLS,X
5300         CMP     ATACHR
5301         BEQ     TSTCT2
5302         DEX
5303         DEX
5304         DEX
5305         BPL     TSTCT1
5306 TSTCT2: RTS
```

```
5307 ;
5308 ;
5309 ;
5310 ; PUSH ROWCRS,COLCRS AND COLCRS+1
5311 ;
5312 PHACRS: LDX     #2
5313 PHACR1: LDA     ROWCRS,X
5314         STA     TMPROW,X
5315         DEX
5316         BPL     PHACR1
5317         RTS
5318 ;
5319 ;
5320 ; PULL COLCRS+1,COLCRS AND ROWCRS
5321 ;
5322 PLACRS: LDX     #2
5323 PLACR1: LDA     TMPROW,X
5324         STA     ROWCRS,X
5325         DEX
5326         BPL     PLACR1
5327         RTS
5328 ;
5329 ;
5330 ;
5331 ; SWAP: IF MIXED MODE, SWAP TEXT CURSORS WITH REGULAR CURSORS
5332 ;
5333 SWAP:   JSR     SWAPA       ;THIS ENTRY POINT DOESRETUR1
5334         JMP     RETURI
5335 SWAPA:  LDA     BOTSCR
5336         CMP     #24
5337         BEQ     SWAP3
5338         LDX     #11
5339 SWAP1:  LDA     ROWCRS,X
5340         PHA
5341         LDA     TXTROW,X
5342         STA     ROWCRS,X
5343         PLA
5344         STA     TXTROW,X
5345         DEX
5346         BPL     SWAP1
5347         LDA     SWPFLG
5348         EOR     #$FF
5349         STA     SWPFLG
5350 SWAP3:  RTS
5351 ;
5352 ;
5353 ; CLICK: MAKE CLICK THROUGH KEYBOARD SPEAKER
5354 ;
5355 CLICK:  LDX     #$7F
5356 CLICK1: STX     CONSOL
5357         STX     WSYNC
5358         DEX
5359         BPL     CLICK1
5360         RTS
5361 ;
5362 ;
5363 ; COLCR:  PUTS EITHER 0 OR LMARQN INTO COLCRS BASED ON MODE AND SWPFLG
5364 ;
5365 COLCR:  LDA     #0
```

```
5366         LDX     SWPFLG
5367         BNE     COLCR1
5368         LDX     DINDEX
5369         BNE     COLCR2
5370 COLCR1: LDA     LMARGN
5371 COLCR2: STA     COLCRS
5372         RTS
5373 ;
5374 ;
5375 ; PUTMSC: PUT SAVMSC INTO ADRESS
5376 ;
5377 PUTMSC: LDA     SAVMSC      ;SETUP ADDRESS
5378         STA     ADRESS
5379         LDA     SAVMSC+1
5380         STA     ADRESS+1
5381         RTS
5382 ;
5383         .PAGE
5384 ;
5385 ;
5386 ; DRAW -- DRAW A LINE FROM OLDROW,OLDCOL TO NEWROW,NEWCOL
5387 ; (THE AL MILLER METHOD FROM BASKETBALL)
5388 DRAW:   LDX     #0
5389         LDA     ICCOMZ      ;TEST COMMAND: $11=DRAW $12=FILL
5390         CMP     #$11
5391         BEQ     DRAWA
5392         CMP     #$12        ;TEST FILL
5393         BEQ     DRAWB       ;YES
5394         LDY     #NVALID     ;NO, SO RETURN INVALID COMMAND
5395         RTS
5396 DRAWB:  INX
5397 DRAWA:  STX     FILFLG
5398         LDA     ROWCRS      ;PUT CURSOR INTO NEWROW,NEWCOL
5399         STA     NEWROW
5400         LDA     COLCRS
5401         STA     NEWCOL
5402         LDA     COLCRS+1
5403         STA     NEWCOL+1
5404         LDA     #1
5405         STA     ROWINC      ;SET UP INITIAL DIRECTIONS
5406         STA     COLINC
5407         SEC
5408         LDA     NEWROW      ;DETERMINE DELTA ROW
5409         SBC     OLDROW
5410         STA     DELTAR
5411         BCS     DRAW1       ;DO DIRECTION AND ABSOLUTE VALUE
5412         LDA     #$FF        ;BORROW WAS ATTEMPTED
5413         STA     ROWINC      ;SET DIRECTION DOWN
5414         LDA     DELTAR
5415         EOR     #$FF        ;DELTAR = |DELTAR|
5416         CLC
5417         ADC     #1
5418         STA     DELTAR
5419 DRAW1:  SEC
5420         LDA     NEWCOL      ;NOW DELTA COLUMN
5421         SBC     OLDCOL
5422         STA     DELTAC
5423         LDA     NEWCOL+1    ;TWO-BYTE QUANTITY
5424         SBC     OLDCOL+1
```

```
5425         STA     DELTAC+1
5426         BCS     DRAW2        ;DIRECTION AND ABSOLUTE VALUE
5427         LDA     #$FF         ;BORROW WAS ATTEMPTED
5428         STA     COLINC       ;SET DIRECTION = LEFT
5429         LDA     DELTAC
5430         EOR     #$FF         ;DELTAC = |DELTAC|
5431         STA     DELTAC
5432         LDA     DELTAC+1
5433         EOR     #$FF
5434         STA     DELTAC+1
5435         INC     DELTAC       ;ADD ONE FOR TWOS COMPLEMENT
5436         BNE     DRAW2
5437         INC     DELTAC+1
5438 DRAW2:  LDX     #2           ;ZERO RAM FOR DRAW LOOP
5439         LDY     #0
5440         STY     COLAC+1
5441 DRAW3A: TYA
5442         STA     ROWAC,X
5443         LDA     OLDROW,X
5444         STA     ROWCRS,X
5445         DEX
5446         BPL     DRAW3A
5447         LDA     DELTAC       ;FIND LARGER ONE (ROW OR COL)
5448 ;       STA     COUNTR       (PREPARE COUNTR AND ENDPT)
5449 ;       STA     ENDPT
5450         INX                  ;MAKE X 0
5451         TAY
5452         LDA     DELTAC+1
5453         STA     COUNTR+1
5454         STA     ENDPT+1
5455         BNE     DRAW3        ;AUTOMATICALLY LARGER IF MSD>0
5456         LDA     DELTAC
5457         CMP     DELTAR       ;LOW COL >LOW ROW?
5458         BCS     DRAW3        ;YES
5459         LDA     DELTAR
5460         LDX     #2
5461         TAY
5462 DRAW3:  TYA                  ;PUT IN INITIAL CONDITIONS
5463         STA     COUNTR
5464         STA     ENDPT
5465         PHA                  ;SAVE AC
5466         LDA     ENDPT+1      ;PUT LSB OF HIGH BYTE
5467         LSR     A            ;INTO CARRY
5468         PLA                  ;RESTORE AC
5469         ROR     A            ;ROR THE 9 BIT ACUMULATOR
5470         STA     ROWAC,X
5471 DRAW4A: LDA     COUNTR       ;TEST ZERO
5472         ORA     COUNTR+1
5473         BNE     DRAWI1       ;IF COUNTER IS ZERO, LEAVE DRAW
5474         JMP     DRAW10
5475 DRAWI1: CLC                  ;ADD ROW TO ROWAC (PLOT LOOP)
5476         LDA     ROWAC
5477         ADC     DELTAR
5478         STA     ROWAC
5479         BCC     DRAW5
5480         INC     ROWAC+1
5481 DRAW5:  LDA     ROWAC+1      ;COMPARE ROW TO ENDPOINT
5482         CMP     ENDPT+1      ;IF HIGH BYTE OF ROW IS .LT. HIGH
5483         BCC     DRAW6        ;BYTE OF ENDPT, BLT TO COLUMN
```

```
5484         BNE     DRAW5A
5485         LDA     ROWAC
5486         CMP     ENDPT        ;LOW BYTE
5487         BCC     DRAW6        ;ALSO TILT
5488 DRAW5A: CLC                  ;GE SO MOVE POINT
5489         LDA     ROWCRS
5490         ADC     ROWINC
5491         STA     ROWCRS
5492         LDX     #0           ;AND SUBTRACT ENDPT FROM ROWAC
5493         JSR     SUBEND
5494 DRAW6:  CLC                  ;DO SAME FOR COLUMN (DOUBLE BYTE ADD)
5495         LDA     COLAC        ;ADD
5496         ADC     DELTAC
5497         STA     COLAC
5498         LDA     COLAC+1
5499         ADC     DELTAC+1
5500         STA     COLAC+1
5501         CMP     ENDPT+1      ;COMPARE HIGH BYTE
5502         BCC     DRAW8
5503         BNE     DRAW6A
5504         LDA     COLAC        ;COMPARE LOW BYTE
5505         CMP     ENDPT
5506         BCC     DRAW8
5507 DRAW6A: BIT     COLINC       ;+ OR - ?
5508         BPL     DRAW6B
5509         DEC     COLCRS       ;DO DOUBLE BYTE DECREMENT
5510         LDA     COLCRS
5511         CMP     #$FF
5512         BNE     DRAW7
5513         LDA     COLCRS+1
5514         BEQ     DRAW7       ;DON'T DEC IF ZERO
5515         DEC     COLCRS+1
5516         BPL     DRAW7        ;(UNCONDITIONAL)
5517 DRAW6B: INC     COLCRS       ;DO DOUBLE BYTE INCREMENT
5518         BNE     DRAW7
5519         INC     COLCRS+1
5520 DRAW7:  LDX     #2           ;AND SUBTRACT ENDPT FROM COLAC
5521         JSR     SUBEND
5522 DRAW8:  JSR     RANGE
5523         JSR     OUTPLT       ;PLOT POINT
5524         LDA     FILFLG       ;TEST RIGHT FILL
5525         BEQ     DRAW9
5526         JSR     PHACRS
5527         LDA     ATACHR
5528         STA     HOLD4
5529 DRAW8A: LDA     ROWCRS       ;SAVE ROW IN CASE OF CR
5530         PHA
5531         JSR     INCRSA       ;POSITION CURSOR ONE PAST DOT
5532         PLA                  ;RESTORE ROWCRS
5533         STA     ROWCRS
5534 DRAW8C: JSR     RANGE
5535         JSR     GETPLT       ;GET DATA
5536         BNE     DRAW8B       ;STOP IF NON-ZERO DATA IS ENCOUNTERED
5537         LDA     FILDAT       ;FILL DATA
5538         STA     ATACHR
5539         JSR     OUTPLT       ;DRAW IT
5540         JMP     DRAW8A       ;LOOP
5541 DRAW8B: LDA     HOLD4
5542         STA     ATACHR
```

```
5543         JSR     PLACRS
5544 DRAW9:  SEC                 ;DO DOUBLE BYTE SUBTRACT
5545         LDA     COUNTR
5546         SBC     #1
5547         STA     COUNTR
5548         LDA     COUNTR+1
5549         SBC     #0
5550         STA     COUNTR+1
5551         BMI     DRAW10
5552         JMP     DRAW4A
5553 DRAW10: JMP     RETUR1
5554         .PAGE
5555 ;
5556 ;
5557 ; TABLES
5558 ;
5559 ;
5560 ; MEMORY ALLOCATION
5561 ;
5562 ALOCAT: .BYTE   24,16,10,10,16,28,52,100,196,196,196,196
5563
5564
5565 ;
5566 ;
5567 ; NUMBER OF DISPLAY LIST ENTRIES
5568 ;
5569 NUMDLE: .BYTE   23,23,11,23,47,47,95,95,97,97,97,97
5570
5571
5572 MXDMDE: .BYTE   19,19,9,19,39,39,79,79,65,65,65,65 ;(EXT OF NUMDLE)
5573
5574
5575 ;
5576 ;
5577 ; ANTIC CODE FROM INTERNAL MODE CONVERSION TABLE
5578 ;
5579 ;   INTERNAL         ANTIC CODE             DESCRIPTION
5580 ;       0               2               40X2X8  CHARACTERS
5581 ;       1               6               20X5X8      ""
5582 ;       2               7               20X5X16     ""
5583 ;       3               8               40X4X8  GRAPHICS
5584 ;       4               9               80X2X4     ""
5585 ;       5               A               80X4X4     ""
5586 ;       4               B               160X2X2    ""
5587 ;       7               D               160X4X2    ""
5588 ;       8               F               320X2X1    ""
5589 ;       9               SAME AS 8 BUT GTIA 'LUM' MODE
5590 ;       10              SAME AS 8 BUT GTIA 'COL/LUM REGISTER' MODE
5591 ;       11              SAME AS 8 BUT GTIA 'COLOR' MODE
5592 ;
5593 ANCONV: .BYTE   2,6,7,8,9,$A,$B,$D,$F,$F,$F,$F ;ZEROS FOR RANGE TEST IN
5594
5595
5596 ;
5597 ;
5598 ; PAGE TABLE TELLS WHICH DISPLAY LISTS ARE IN DANGER OF
5599 ; CROSSING A 256 BYTE PAGE BOUNDARY
5600 ;
5601 PAGETB: .BYTE   0,0,0,0,0,0,0,1,1,1,1,1
```

```
5602
5603
5604 ;
5605 ;
5606 ; THIS IS THE NUMBER OF LEFT SHIFTS NEEDED TO MULTIPLY
5607 ; COLCRS BY 10,20, OR 40.  (ROWCRS*10)/(2**DHLINE)
5608 ;
5609 DHLINE: .BYTE   2,1,1,0,0,1,1,2,2,2,2,2
5610
5611
5612 ;
5613 ;
5614 ; COLUMN: NUMBER OF COLUMNS
5615 ;
5616 COLUMN: .BYTE   40,20,20,40,80,80,160,160,64,80,80,80 ;MODE 8 IS SPECIAL
5617
5618
5619 ;
5620 ;
5621 ;
5622 ; NOROWS: NUMBER OF ROWS
5623 ;
5624 NOROWS: .BYTE   24,24,12,24,48,48,96,96,192,192,192,192
5625
5626
5627 ;
5628 ;
5629 ;
5630 ;
5631 ; DIV2TB: HOW MANY RIGHT SHIFTS FOR HCRSR FOR PARTIAL BYTE MODES
5632 ;
5633 DIV2TB: .BYTE   0,0,0,2,3,2,3,2,3,1,1,1
5634
5635
5636 ;
5637 ;
5638 ; DMASKT: DISPLAY MASK TABLE
5639 ;
5640 DMASKT: .BYTE   $00,$FF,$F0,$0F
5641         .BYTE   $C0,$30,$0C,$03
5642 ;
5643 ; MASKTB: BIT MASK.   (ALSO PART OF DMASKTB DO NOT SEPARATE)
5644 ;
5645 MASKTB: .BYTE   $80,$40,$20,$10,$08,$04,$02,$01
5646
5647 ;
5648 ;
5649 ;
5650 ;
5651 COLRTB: .BYTE   $28,$CA,$94,$46,$00
5652
5653 ;
5654 ;
5655 ;
5656 ;
5657 ;CNTRLS: CONTROL CODES AND THEIR DISPLACEMENTS INTO THE
5658 ;        CONTROL CHARACTER PROCESSORS
5659 ;
5660 CNTRLS: .BYTE   $1B
```

```
5661         .WORD   ESCAPE
5662         .BYTE   $1C
5663         .WORD   CRSRUP
5664         .BYTE   $1D
5665         .WORD   CRSRDN
5666         .BYTE   $1E
5667         .WORD   CRSRLF
5668         .BYTE   $1F
5669         .WORD   CRSRRT
5670         .BYTE   $7D
5671         .WORD   CLRSCR
5672         .BYTE   $7E
5673         .WORD   BS
5674         .BYTE   $7F
5675         .WORD   TAB
5676         .BYTE   $9B
5677         .WORD   DOCRWS
5678         .BYTE   $9C
5679         .WORD   DELLIN
5680         .BYTE   $9D
5681         .WORD   INSLIN
5682         .BYTE   $9E
5683         .WORD   CLRTAB
5684         .BYTE   $9F
5685         .WORD   SETTAB
5686         .BYTE   $FD
5687         .WORD   BELL
5688         .BYTE   $FE
5689         .WORD   DELCHR
5690         .BYTE   $FF
5691         .WORD   INSCHR
5692 ;
5693 ;
5694 ;
5695 ;
5696 ;
5697 ; ATAINT: ATASCI TO INTERNAL TABLE
5698 ;
5699 ATAINT: .BYTE   $40,$00,$20,$60
5700 ;
5701 ;
5702 ; INTATA: INTERNAL TO ATASCI TABLE
5703 ;
5704 INTATA: .BYTE   $20,$40,$00,$60
5705 ;
5706 ;
5707 ; ATASCI: ATASCII CONVERSION TABLE
5708 ;
5709 ATASCI: .BYTE   $6C,$6A,$3B,$80,$80,$6B,$2B,$2A ;LOWER CASE
5710
5711         .BYTE   $6F,$80,$70,$75,$9B,$69,$2D,$3D
5712
5713
5714         .BYTE   $76,$80,$63,$80,$80,$62,$78,$7A
5715
5716         .BYTE   $34,$80,$33,$36,$1B,$35,$32,$31
5717
5718
5719         .BYTE   $2C,$20,$2E,$6E,$80,$6D,$2F,$81
```

```
5720
5721         .BYTE   $72,$80,$65,$79,$7F,$74,$77,$71
5722
5723
5724         .BYTE   $39,$80,$30,$37,$7E,$38,$3C,$3E
5725
5726         .BYTE   $66,$68,$64,$80,$82,$67,$73,$61
5727
5728
5729
5730         .BYTE   $4C,$4A,$3A,$80,$80,$4B,$5C,$5E ;UPPER CASE
5731
5732         .BYTE   $4F,$80,$50,$55,$9B,$49,$5F,$7C
5733
5734
5735         .BYTE   $56,$80,$43,$80,$80,$42,$58,$5A
5736
5737         .BYTE   $24,$80,$23,$26,$1B,$25,$22,$21
5738
5739
5740         .BYTE   $5B,$20,$5D,$4E,$80,$4D,$3F,$81
5741
5742         .BYTE   $52,$80,$45,$59,$9F,$54,$57,$51
5743
5744
5745         .BYTE   $28,$80,$29,$27,$9C,$40,$7D,$9D
5746
5747         .BYTE   $46,$48,$44,$80,$83,$47,$53,$41
5748
5749
5750
5751         .BYTE   $0C,$0A,$7B,$80,$80,$0B,$1E,$1F ;CONTROL
5752
5753         .BYTE   $0F,$80,$10,$15,$9B,$09,$1C,$1D
5754
5755
5756         .BYTE   $16,$80,$03,$80,$80,$02,$18,$1A
5757
5758         .BYTE   $80,$80,$85,$80,$1B,$80,$FD,$80
5759
5760
5761         .BYTE   $00,$20,$60,$0E,$80,$0D,$80,$81
5762
5763         .BYTE   $12,$80,$05,$19,$9E,$14,$17,$11
5764
5765
5766         .BYTE   $80,$80,$80,$80,$FE,$80,$7D,$FF
5767
5768         .BYTE   $06,$08,$04,$80,$84,$07,$13,$01
5769
5770 ;
5771 ;
5772 ;
5773 ;
5774 ;
5775 PIRQ5:  LDA     KBCODE
5776         CMP     CH1          ;TEST AGAINST LAST KEY PRESSED
5777         BNE     PIRQ3        ;IF NOT, GO PROCESS KEY
5778         LDA     KEYDEL       ;IF KEY DELAY BYTE > 0
```

```
5779         BNE     PIRQ4       ;IGNORE KEY AS BOUNCE
5780 PIRQ3:  LDA     KBCODE      ;RESTORE AC
5781         CMP     #CNTL1      ;TEST CONTROL 1 (SSFLAG)
5782         BNE     PIRQ1
5783         LDA     SSFLAG
5784         EOR     #$FF
5785         STA     SSFLAG
5786         BCS     PIRQ4       ;(UNCONDITIONAL) MAKE ^1 INVISIBLE
5787 PIRQ1:  STA     CH
5788         STA     CH1
5789         LDA     #3
5790         STA     KEYDEL      ;INITIALIZE KEY DELAY FOR DEBOUNCE
5791         LDA     #0          ;CLEAR COLOR SHIFT BYTE
5792         STA     ATRACT
5793 PIRQ4:  LDA     #$30
5794         STA     SRTIMR
5795 PIRQ2:  PLA
5796         RTI
5797 ;
5798 ;
5799         .BYTE   $FF,$FF,$FF,$FF,$FF,$FF
5800
5801 ;
5802 CRNTPC  =*
5803         *=$14
5804 KBDSPR: .BYTE   $FFF8-CRNTPC ;^GDISPLC IS TOO LONG
5805         .END
5806
```